

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Eksamenskurs
 Moro: 2D-Nim
 Spørsmål

Dynamisk programmering

Åsmund Eldhuset

asmunde *at* stud.ntnu.no
 folk.ntnu.no/asmunde/algdat/dp.ppt

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

Svært rask repetisjon

- Noen ganger (f.eks. ved utregning av Fibonaccitall) vil en rekursiv funksjon få eksponentiell kjøretid fordi den regner ut de samme delsvarene gang på gang
- Memoisering: få den rekursive funksjonen til å mellomlagre svarene i et dictionary eller et array (går ovenfra og ned)
- Dynamisk programmering: droppe rekursjonen og fyller ut tabellen direkte (går nedenfra og opp)

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

Rekursjonstre – Fibonacci-tall

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

Rekursiv problemformulering

- Det gjelder å formulere løsningen av et problem som en kombinasjon av løsninger av mindre utgaver av *samme* problem
- Eksempel: Fibonacci-tall nummer n er summen av Fibonacci-tall nummer $n - 1$ og Fibonacci-tall nummer $n - 2$
- De mindre utgavene kalles delproblemer

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

Longest increasing subsequence (LIS)

- Problem: Gitt en rekke med n reelle tall $a_1, a_2, a_3, \dots, a_n$, finn den største delrekken slik at tallene står i stigende rekkefølge
- Eksempel: [1, 2, 9, 3, 8, 5, 7]
- Fungerer en grådig algoritme?
- Nei, fordi delproblemene ikke er uavhengige. Velger man å ta med et bestemt tall, har man lagt begrensninger på hvilke tall man kan få med seg senere

:: Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – optimal substruktur

- Vi observerer at en LIS for en bestemt liste består av en LIS for en mindre del av listen
- Eksempel: [1, 2, 9, 3, 8, 5, 7]
- LIS'en er [1, 2, 3, 5, 7]
- [1, 2, 3, 5] er en LIS for [1, 2, 9, 3, 8, 5]
- [1, 2, 3] er en LIS for [1, 2, 9, 3]
- [1, 2] er en LIS for [1, 2]
- [1] er en LIS for [1]
- Konklusjon: en optimal løsning av problemet består av optimale løsninger av delproblemer

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – rekursiv definisjon

- La oss se på problemet med å finne en LIS som slutter med tall nummer i , og la oss kalle lengden av dette for $L(i)$
- Da vil hovedproblemet vårt være å finne det beste av $L(1), L(2), L(3), \dots, L(n)$
- Hvordan regner vi ut $L(i)$?

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – rekursiv definisjon

- Vi finner lengste delsekvensen som ikke inneholder tall større enn tall nummer i
- Altså: vi må finne den j som er slik at $j < i$, $a_j \leq a_i$, og $L(j)$ er størst mulig
- $L(i)$ er da lik den største $L(j)$ pluss 1
- $L(i) = \max(L(j) : 0 \leq j < i \wedge a_j \leq a_i) + 1$
- $L(0) = 0$

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – tabellutfylling

```

[1, 2, 9, 3, 8, 5, 7]
[0, -, -, -, -, -, -]
[0, 1, -, -, -, -, -]
[0, 1, 2, -, -, -, -]
[0, 1, 2, 3, -, -, -]
[0, 1, 2, 3, 3, -, -]
[0, 1, 2, 3, 3, 4, -]
[0, 1, 2, 3, 3, 4, 4, -]
[0, 1, 2, 3, 3, 4, 4, 5]

```

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – sporing

- Nå har vi funnet lengden på LIS'en, men det hadde jo vært kjekt å finne selve LIS'en også...
- I et array P registrerer vi hvilket delproblem hvert delproblem benyttet
- a : [1, 2, 9, 3, 8, 5, 7]
 L : [0, 1, 2, 3, 3, 4, 4, 5]
 P : [-, 0, 1, 2, 2, 4, 4, 6]

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – implementasjon

```

def LIS(a):
    n = len(a)
    L = [0] * (n + 1)
    P = [-1] * (n + 1)
    for i in range(1, n + 1):
        bestIndex = 0
        for j in range(1, i):
            if a[j - 1] <= a[i - 1] and \
L[j] > L[bestIndex]:
                bestIndex = j
            L[i] = L[bestIndex] + 1
            P[i] = bestIndex
    return L[n], P

```

Forside
 Repetisjon
 Longest increasing subsequence
 Betingelser
 Matrise-multiplikasjon
 Longest common subsequence
 Knapsack
 Grådig vs. DP
 Moro: 2D-Nim
 Spørsmål

LIS – implementasjon

```

def makeLIS(a, P, i):
    if i == 0:
        return []
    else:
        b = makeLIS(a, P, P[i])
        b.append(a[i - 1])
        return b

def makeLIS(a, P):
    i = len(a)
    b = []
    while i != 0:
        b.append(a[i - 1])
        i = P[i]
    b.reverse()
    return b

```

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Betingelse 1: Optimal substruktur

- Ofte har vi en situasjon hvor vi ønsker å finne den best mulige løsningen av et problem
- Skal vi kunne bruke DP, må det være slik at problemet har en *optimal substruktur* – en optimal løsning av hele problemet består av optimale løsninger av delproblemer
- Når vi løser et delproblem, må vi som regel se gjennom alle de mindre delproblemene og velge det beste

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Betingelse 1.5: Uavhengige delproblemer

- Løsningen av ett delproblem må ikke legge begrensninger på hvordan andre delproblemer kan løses
- Delproblemer må ikke ha felles ressurser som kan spises opp
- (Anti)eksempel: Longest simple path
 - En LSP fra a til b må bestå av en LSP fra a til x og fra x til b (vi har optimal substruktur)
 - Men når vi finner en LSP fra a til x , risikerer vi å bruke opp noder som LSP fra x til b kunne ha brukt (vi har ikke uavhengige delproblemer)

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Betingelse 2: Overlappende delproblemer

- Det burde heller hete "gjentatte delproblemer"
- Løsningen av to forskjellige delproblemer kan involvere løsning av ett eller flere felles delproblemer
- Derfor blir rekursjon uten memoisering ekstremt ineffektivt fordi det samme arbeidet gjøres gang på gang
- Dersom delproblemene ikke overlapper, kan en splitt-og-hersk-algoritme brukes

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

(Betingelse 3: Antall delproblemer)

- Det er bare vits i DP hvis antallet delproblemer er "lite" nok til at vi kan lagre alle svarene i minnet og får løst alle delproblemene raskt nok

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Ting å være oppmerksom på

- DP fungerer som regel bare på problemer der ting må komme i en bestemt rekkefølge, eller hvor du står fritt til å plukke en rekkefølge selv (slik som Floyd-Warshall gjør)
- Selv om DP tilsynelatende gir polynomisk kjøretid, kan det fremdeles være eksponentielt i inputstørrelsen! (f.eks. myntutdeling eller 0/1 knapsack)

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Matrisemultiplikasjon

- Eksempel: Regne ut $A_1 A_2 A_3 A_4 A_5$
- Dimensjonene må passe ($a \times b$, $b \times c$, $c \times d$, $d \times e$, $e \times f$)
- Matrisemultiplikasjon er ikke kommutativ (du kan ikke bytte om på rekkefølgen)
- Men det er assosiativt (du kan velge i hvilken rekkefølge du utfører multiplikasjonene)
- Vi kan kontrollere multiplikasjonsrekkefølgen ved å sette på parenteser:

$$A_1 A_2 A_3 A_4 A_5 = A_1 ((A_2 A_3) (A_4 A_5)) = ((A_1 A_2) A_3) (A_4 A_5) = A_1 (A_2 (A_3 (A_4 A_5))) = \dots$$

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Matrisemultiplikasjon

- Multiplikasjon av to matriser med dimensjoner $a \times b$ og $b \times c$ gir abc skalar-multiplikasjoner (multiplikasjoner av tall)
- Forskjellige parentessettinger fører som regel til forskjellig totalt antall skalarmultiplikasjoner
- Hvilken parentessetting er best?

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Matrisemultiplikasjon

- Fullt parentessatt uttrykk: enten en enkelt matrise eller et produkt av to uttrykk som er fullt parentessatt
- Alle måter å multiplisere sammen matriser på kan skrives fullt parentessatt
- Den beste måten må derfor være et fullt parentessatt uttrykk – altså et produkt av to andre FPU, som selv må være optimale

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Longest common subsequence (LCS)

- Finne lengste tegnsekvens som er felles for to strenger
- Eksempel:
A = "abqaeehgpcydkpk1z"
B = "rwazzbaertctdz"
- Vi definerer $LCS(i, j)$ til å være LCS'en til $A[0:i]$ og $B[0:j]$

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Longest common subsequence (LCS)

- Dersom siste tegn i $A[0:i]$ og $B[0:j]$ er like, er disse en del av $LCS(i, j)$; da trenger vi bare å finne $LCS(i-1, j-1)$ og legge til 1
- Hvis ikke, kan ikke begge de siste tegnene være med i $LCS(i, j)$
- Da er $LCS(i, j)$ enten lik $LCS(i-1, j)$ eller $LCS(i, j-1)$
- Vi får da en tabell over $LCS(i, j)$ på $m \times n$, hvor m er lengden til A og n er lengden til B

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Fractional knapsack

- Kan ikke løses med DP, siden antallet forskjellige delproblemer blir uendelig!
- DP krever som regel løsning av alle delproblemer for å kunne velge det best egnede, mens en grådig algoritme gjør et grådig valg og produserer ett delproblem

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Grådig vs. DP

- Begge utnytter optimal substruktur
- Grådig
 - Lokalt optimale valg er globalt optimale; optimal løsning avgjøres ut fra hva som virker best der og da
 - Løses som regel ovenfra og ned; man tar et valg og ender opp med et mindre delproblem
- DP
 - Optimal løsning avgjøres ved å se på optimale løsninger av delproblemer
 - Løses nedenfra og opp; man løser større og større delproblemer ut i fra de mindre del løsningene

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Sammendrag: krav for DP

- Rekursiv struktur
- Optimal substruktur
- Overlappende delproblemer (hvis ikke: splitt og hersk-algoritme)
- Løsning av ett delproblem avhenger av løsningene av andre delproblemer (hvis ikke: grådig algoritme)

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Faglig underholdning: Nim i to dimensjoner

- Nim er et fyrstikkspill der man har en haug fyrstikker og trekker 1-3 fyrstikker annenhver gang (antallet kan variere) og den som tar den siste fyrstikken vinner
- Vi vil nå spille Nim i to dimensjoner: vi har et ruteark, og man skal hver gang klippe bort en remse på 1-3 ruters bredde fra en av kantene. Den som får servert en 1x1-rute til slutt har tapt
- Er det mulig for den som begynner å følge en strategi slik at han garantert vinner?

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Faglig underholdning: Nim i to dimensjoner

- En tilstand i spillet er beskrevet av størrelsen til det gjenværende rutearket
- En posisjon er vinnende hvis det er mulig å gjøre et trekk slik at motstanderen får servert en tapende posisjon
- En tapende posisjon er enten en posisjon hvor man taper i følge reglene, eller en posisjon hvor man ikke kan sette motstanderen i en annen tapende posisjon
- La oss sette opp dette i en tabell

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Faglig underholdning: Nim i to dimensjoner

	1	2	3	4	5	6	7	8	9
1	L								
2									
3									
4									
5									
6									
7									
8									
9									

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Faglig underholdning: Nim i to dimensjoner

	1	2	3	4	5	6	7	8	9
1	L	W	W	W	L	W	W	W	L
2									
3									
4									
5									
6									
7									
8									
9									

Forside
Repetisjon
Longest increasing subsequence
Betingelser
Matrise-multiplikasjon
Longest common subsequence
Knapsack
Grådig vs. DP
Moro: 2D-Nim
Spørsmål

Faglig underholdning: Nim i to dimensjoner

	1	2	3	4	5	6	7	8	9
1	L	W	W	W	L	W	W	W	L
2	W	L	W	W	W	L	W	W	W
3									
4									
5									
6									
7									
8									
9									

Faglig underholdning:
Nim i to dimensjoner

	1	2	3	4	5	6	7	8	9
1	L	W	W	W	L	W	W	W	L
2	W	L	W	W	W	L	W	W	W
3	W	W	L	W	W	W	L	W	W
4	W	W	W	L	W	W	W	L	W
5	L	W	W	W	L	W	W	W	L
6	W	L	W	W	W	L	W	W	W
7	W	W	L	W	W	W	L	W	W
8	W	W	W	L	W	W	W	L	W
9	L	W	W	W	L	W	W	W	L

Faglig underholdning:
Nim i to dimensjoner

- Dermed kan man gjøre optimale trekk slik:

```

mod = (height - width) % (maxCut + 1)
if mod == 0:
    if width < height:
        height -= 1
    else:
        width -= 1
else:
    if mod < height:
        height -= mod
    else:
        width -= maxCut + 1 - mod
return width, height

```

Spørsmål?