

Grådighet

Øvingsforelesning

TDT4120 Algoritmer og Datastrukturer

Simon Jonassen



Grådighet

greedy - adjective

1. immoderately desirous of acquiring e.g. wealth; "they are avaricious and will do anything for money"; "casting covetous eyes on his neighbor's fields"; "a grasping old miser"; "grasping commercialism"; "greedy for money and power"; "grew richer and greedier"; "prehensile employers stingy with raises for their employees" [syn: avaricious]
2. (often followed by 'for') ardently or excessively desirous; "avid for adventure"; "an avid ambition to succeed"; "fierce devouring affection"; "the esurient eyes of an avid curiosity"; "greedy for fame" [syn: avid]
3. wanting to eat or drink more than one can reasonably consume; "don't be greedy with the cookies"

WordNet® 3.0, © 2006 by Princeton University.

Grådighet

- Grådighet er (Bl.a.) en generell teknikk for optimeringsproblemer
- Vi prøver oss på noe lurt og satser på at dette er riktig!
 - Hva som er lurt?
 - Blir dette riktig?




Grådighetsegenskaper


- Optimal delstruktur
 - en optimal løsning til et problem inneholder optimale løsninger til delproblemene
- 'The Greedy Choice Property'
 - en globalt optimal løsning kan oppnås ved å ta lokalt optimale beslutninger

OBS! Disse skal bevises (evt. på eksamen).

Grådighet, tilleggskrav

- Minst et av delprobleme må kunne løses for seg selv
 - Det er ikke lov å gjøre noe som strider med betingelsene til problemet
 - Det er ikke lov å angre noe vi har gjort tidligere.
- 

Activity Selection Problem

- Gitt flere aktiviteter med en start-tid $S[i]$ og en slutt-tid $F[i]$, finn et maksimalt antall av ikke-koliderende aktiviteter.
 - Grådighet:
 - Sorterer aktivitetene etter sluttid og velger den første først.
 - For alle de gjenstående, hvis denne aktiviteten kolliderer med den forrige vi har tatt, forkaster den. Tar den med ellers.
- 

Activity Selection Problem

- Eksempel:

http://www.cs.fsu.edu/~cop4531/slideshow/images_content/figure17_1.gif


- Lurespørsmål:

- Gitt flere aktiviteter med en start-tid $S[i]$, en slutt-tid $F[i]$ og en belønning $M[i]$, finn et utvalg av ikke-koliderende aktiviteter som gir en maksimal belønning.

(Hint: Grådighet? DP?)




Eksempelproblemer

- Activity Selection Problem (forelest tidligere)
 - Fractional Knapsack Problem (forelest tidligere)
 - OBS: virker ikke på 0-1 Knapsack Problem
 - Shortest Path (forelest tidligere)
 - Minimum Spanning Tree (forelest tidligere)
 - Optimale Prefikskoder (kommer nu!)
 - Approksimasjonsløsninger til Traveling Salesperson Problem, Set Covering, osv. (akvk)
- 

Huffman Codes

- Prefikskoding:
 - ingen gyldig kodelengde er et prefiks i et annet gyldig kodelengde.
 - $\{0, 10, 110, 1110, 11110, \dots\}$, men også $\{00, 01, 10, 11\}$
- Gitt en frekvens $f(c)$ og en lengde $d(c)$ for hver streng c i en koding C . Den gjennomsnittlige bitlengden til en streng i C er da den totale summen av $d(c) * f(c)$
- Et optimalt prefikstre er et fullt binært tre!

Huffman Codes

- Har en mengde av strenger of frekvenser. Lager en node for hver streng vi har.
 - Så lenge det er to eller flere noder igjen:
 - fjerner to noder med lavest frekvens
 - lager en foreldernode med frekvensverdien lik summen av frekvensene til disse to
 - legger foreldernoden tilbake til de gjenstående nodene.
 - Teoremene i Boken Beviser at dette resulterer i et fullt Binært tre som gir en optimal prefikskode.
- 

Huffman, eksempel

1

2

4

5

6

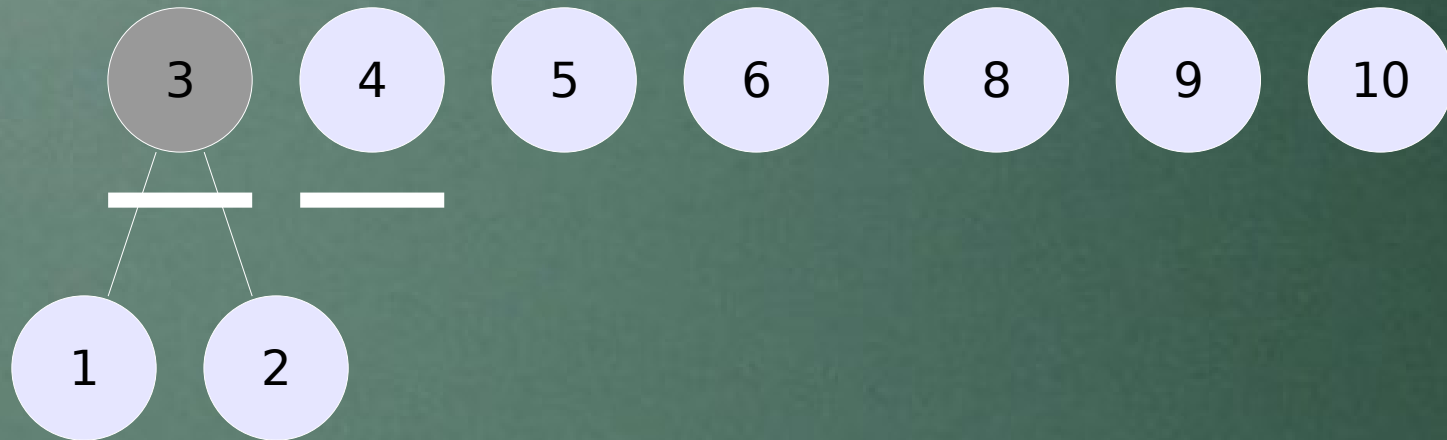
8

9

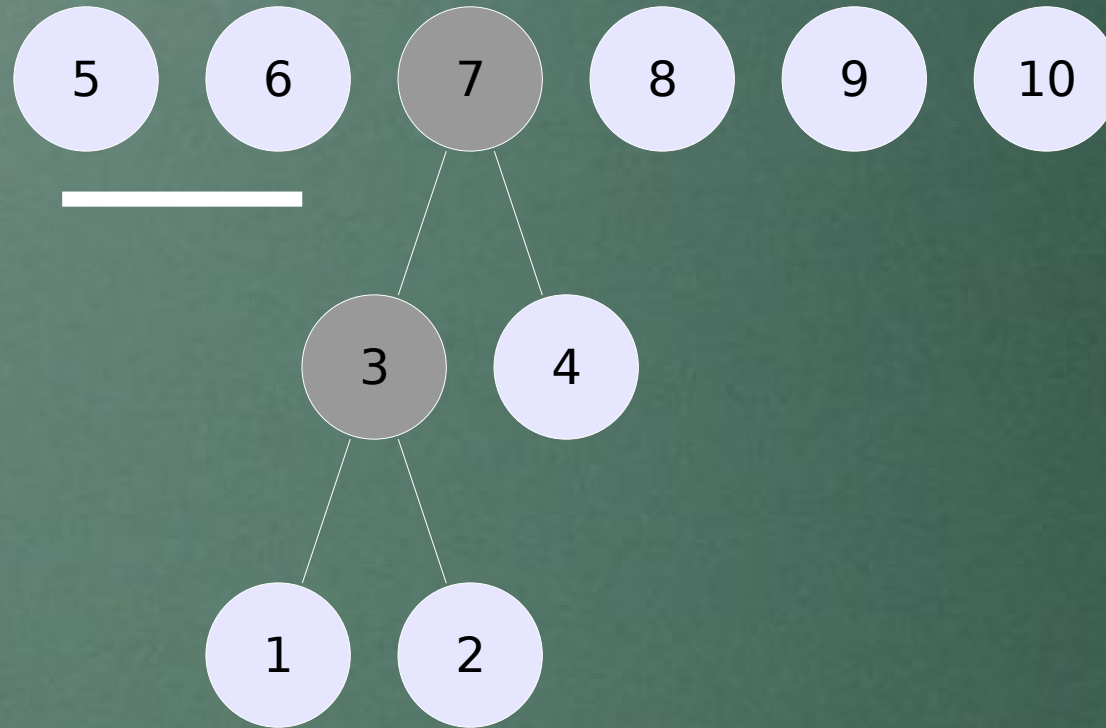
10



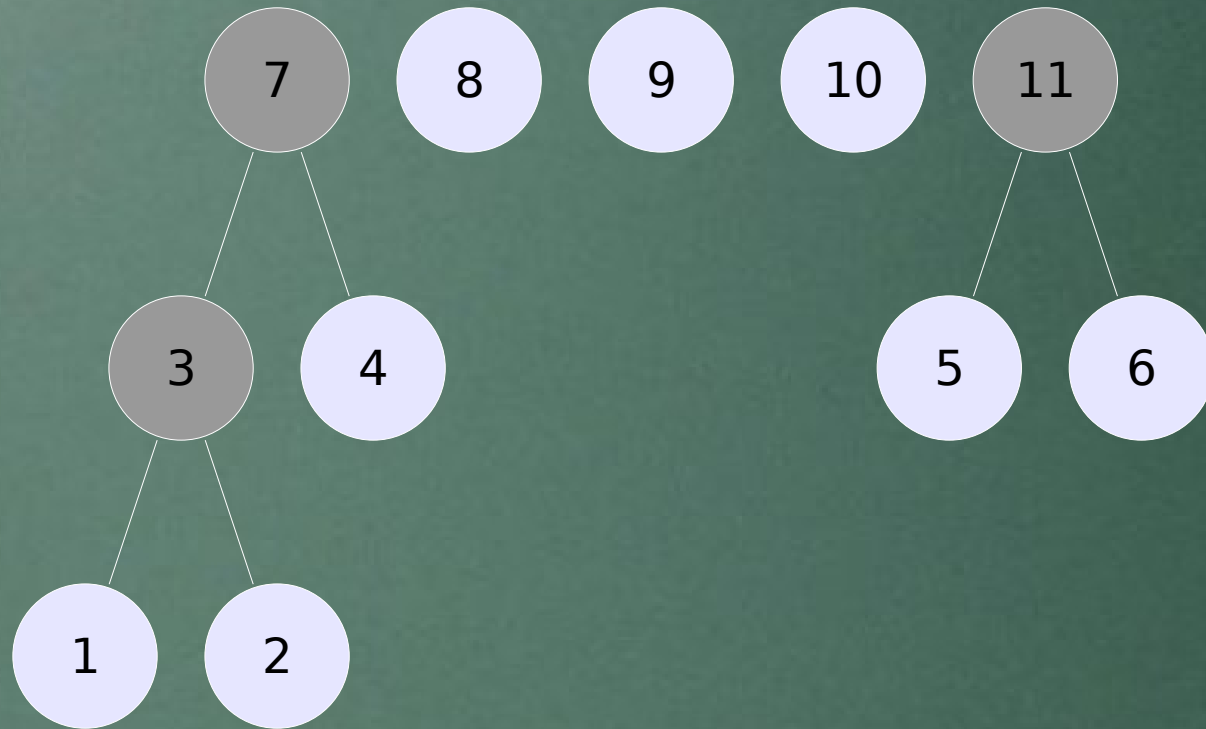
Huffman, eksempel



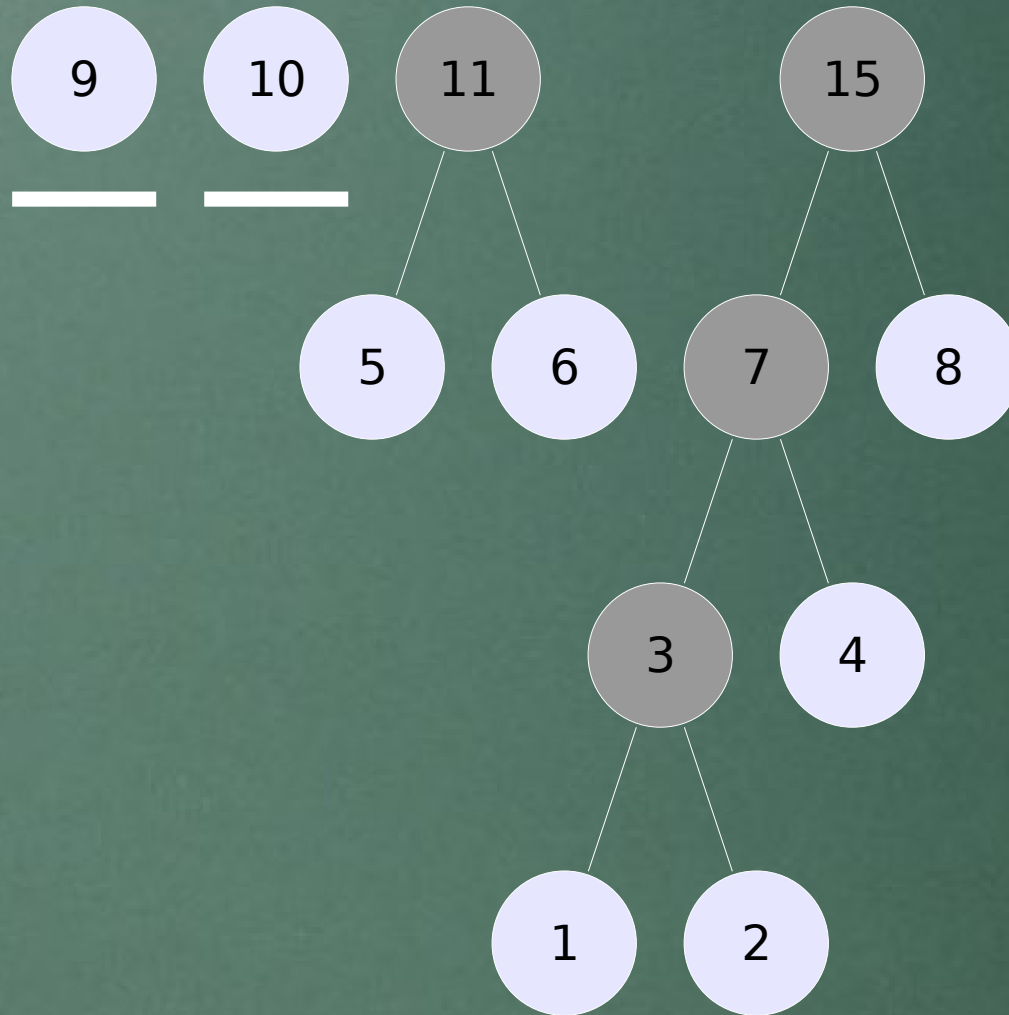
Huffman, eksempel



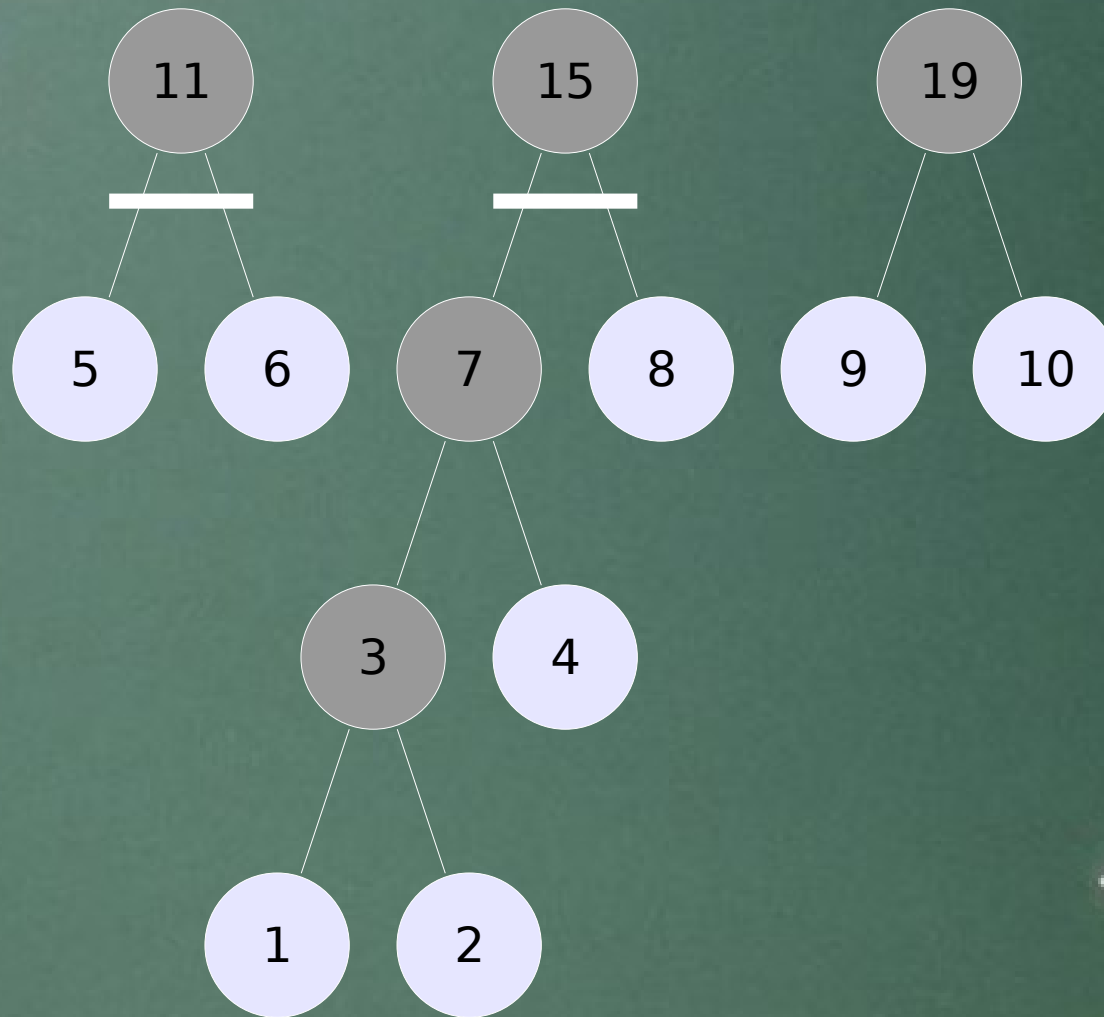
Huffman, eksempel



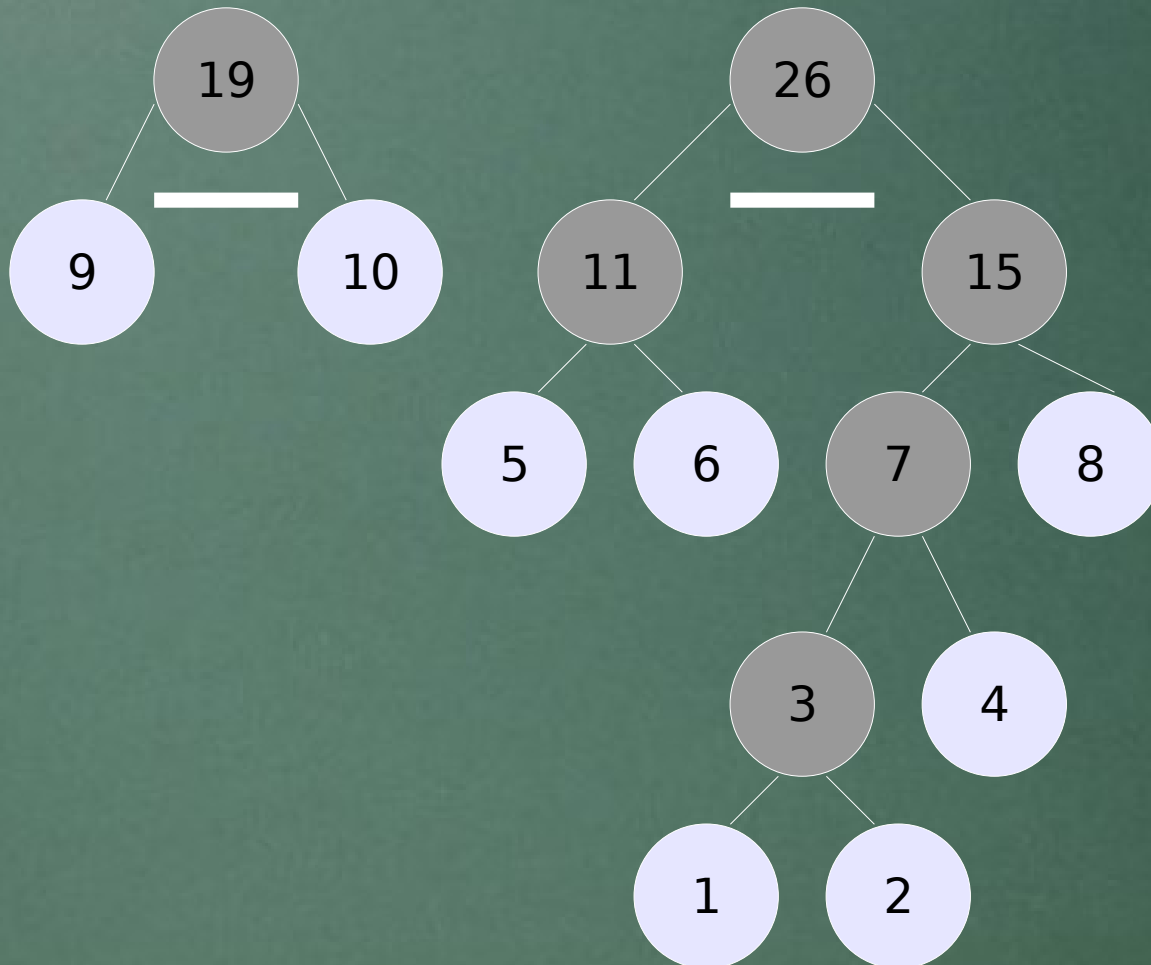
Huffman, eksempel



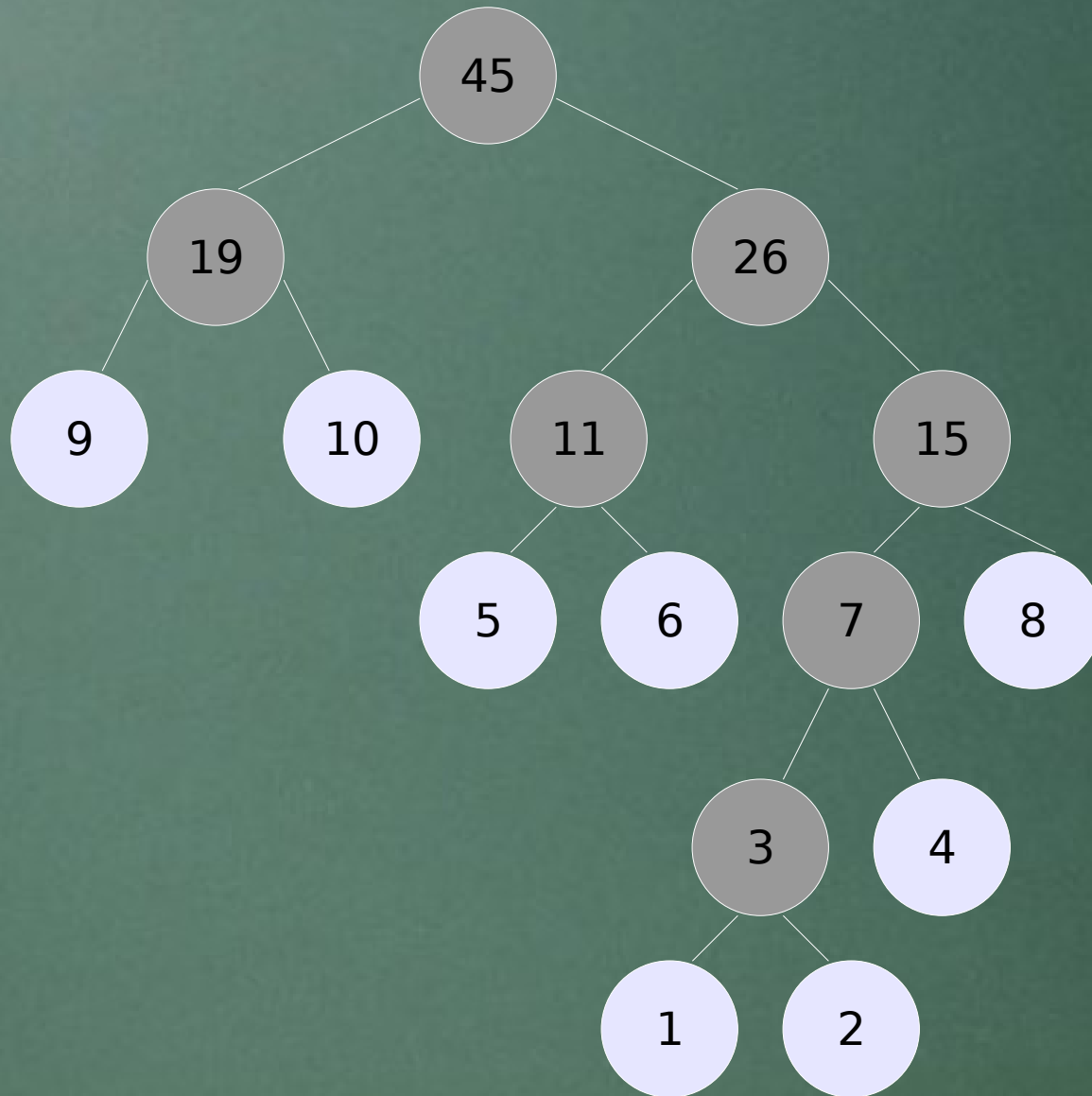
Huffman, eksempel



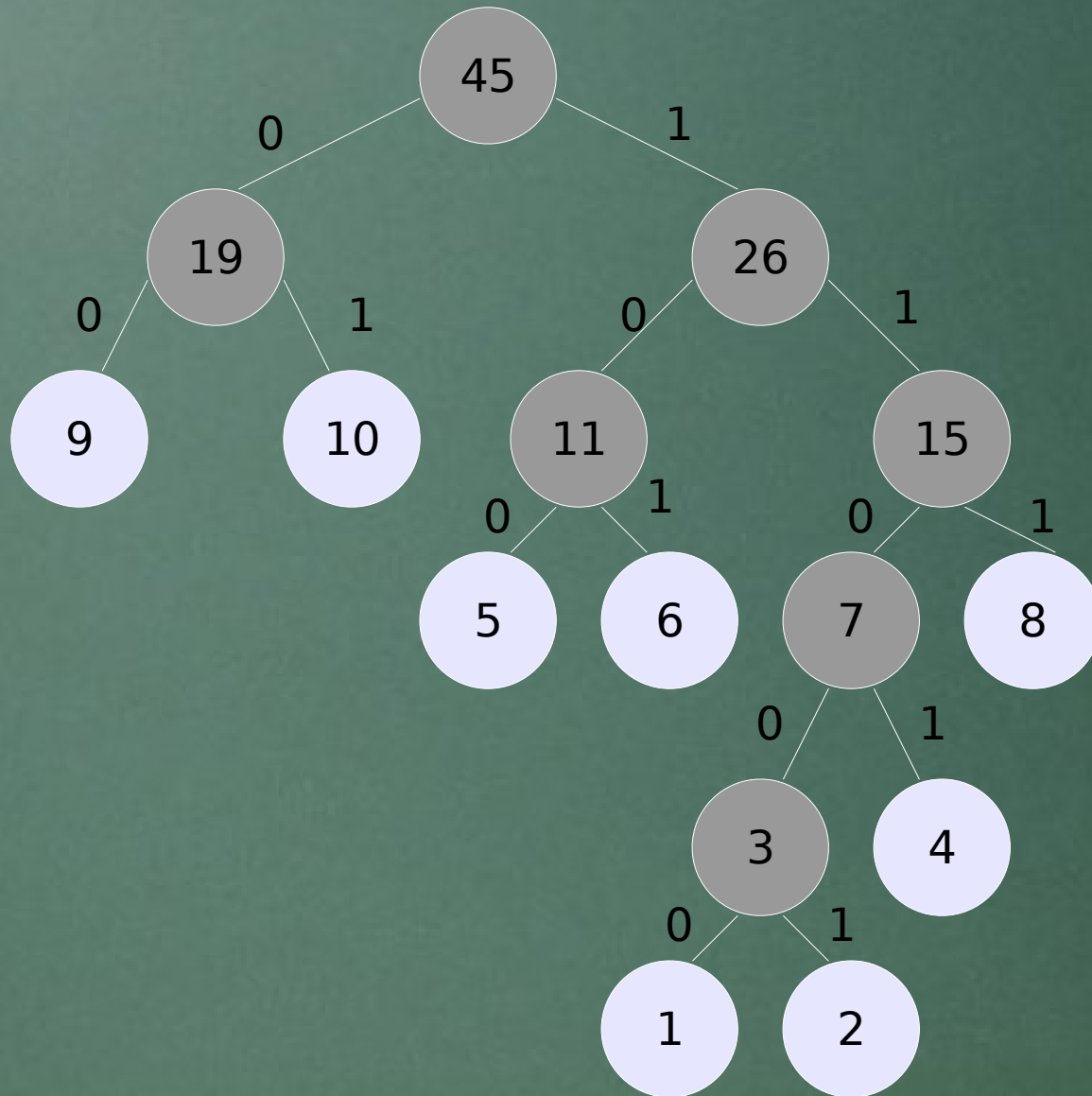
Huffman, eksempel




Huffman, eksempel



Huffman, eksempel



Huffman, eksempel

- $B(\text{Fast}) = 3 * 45 = 135$
 - 3 Bits per symbol
 - 45 er summen av frekvensene
 - $B(\text{Huffman}) = 126$
 - $B(\text{Fast}) - B(\text{Huffman}) = 9$
 - Kan man bruke Huffmans prinsippet til noe annet enn koding?
- 

En eksamensoppgave fra 2004

Oppgave 6 (15%)

Student Lurvik skal summere n positive flyttall, $x[1] \dots x[n]$, slik at avrundingsfeilen blir minst mulig. Avhengig av hvordan dette gjøres, kan de ulike tallene delta i et ulikt antall summeringer. For eksempel vil $x[1]$, $x[2]$, $x[3]$ og $x[5]$ delta i flere summeringer (tre) enn $x[6]$ (én) i følgende summeringsrekkefølge:

$$(((x[1] + x[3]) + (x[2] + x[5])) + x[6])$$

Merk her at både tall-rekkefølgen og parentes-settingen velges fritt.

Hvor stor avrundingsfeilen for en sum av to flyttall blir er avhengig av hvor stor summen er (større sum gir større feil). Lurvik innser at det kan være lurt å summere slik at de små flyttallene deltar i flest mulig summeringer (det vil si, de kommer "dypt" i parentes-settingen) mens de store flyttallene deltar i færrest mulig. Hvis $p(i)$ er "parentes-dybden" til flyttall $x[i]$ (antall parenteser utenfor elementet) så definerer Lurvik *feilen* til en mulig løsning som

$$p(1) \cdot x[1] + p(2) \cdot x[2] + \dots + p(n) \cdot x[n].$$

Lurvik ønsker nå å finne en parentes-setting som minimaliserer dette feiluttrykket. Skisser en løsning på problemet. Hvilken designmetode bruker du? Vis, med støtte i pensum, at løsningen er korrekt.