Grådige algoritmer

Plan:

- Introduktion
  - Hvad er grådige algoritmer?
  - Opvarmning: Eksempel med veksling af mønter
- Aktivitetsudvælgelse (Activity selection)
- Datakompression og Huffmann-koder

Mikkel Abrahamsen

Veksling af mønter

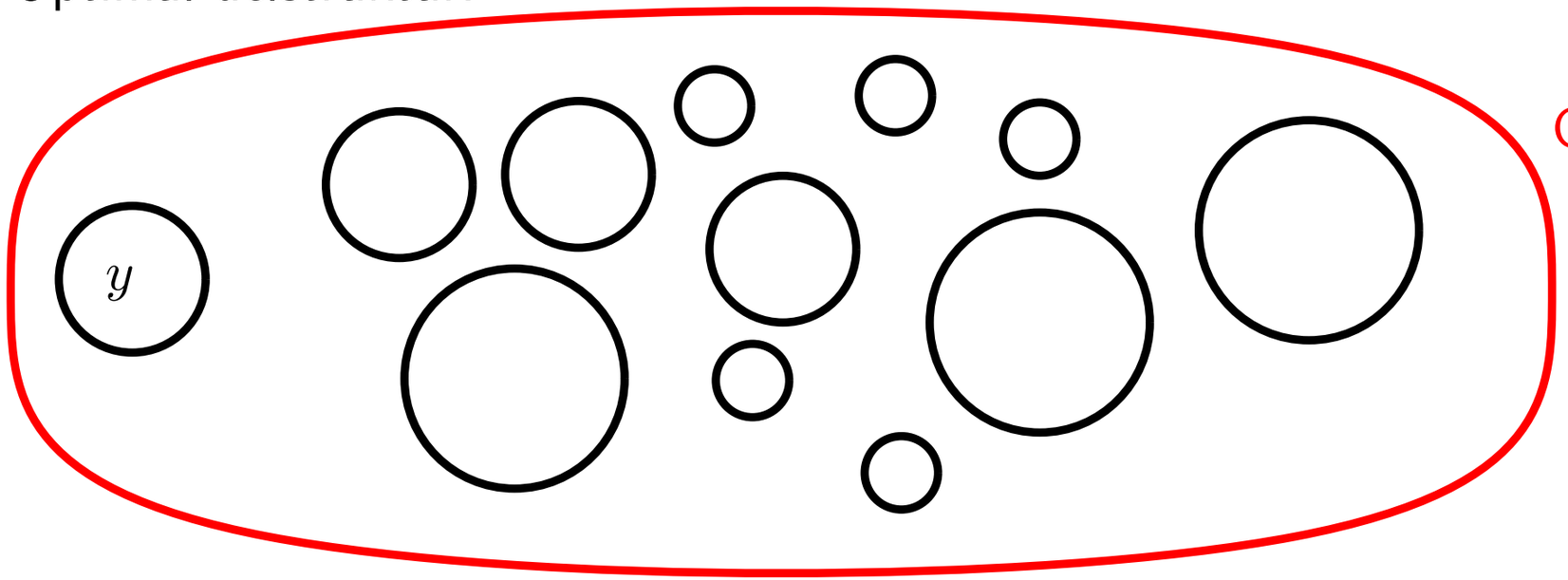
Problem: Givet beløb x, find minimum antal mønter hvis sum er x.
Find a_5, a_2, a_1 så $5a_5 + 2a_2 + a_1 = x$ og minimér $a_5 + a_2 + a_1$.

Grådig algoritme: Brug største mønt som er højst x og fortsæt på samme måde med restbeløbet.

$4 = 2 + 2$
$18 = 5 + 5 + 5 + 2 + 1$

Veksling af mønter

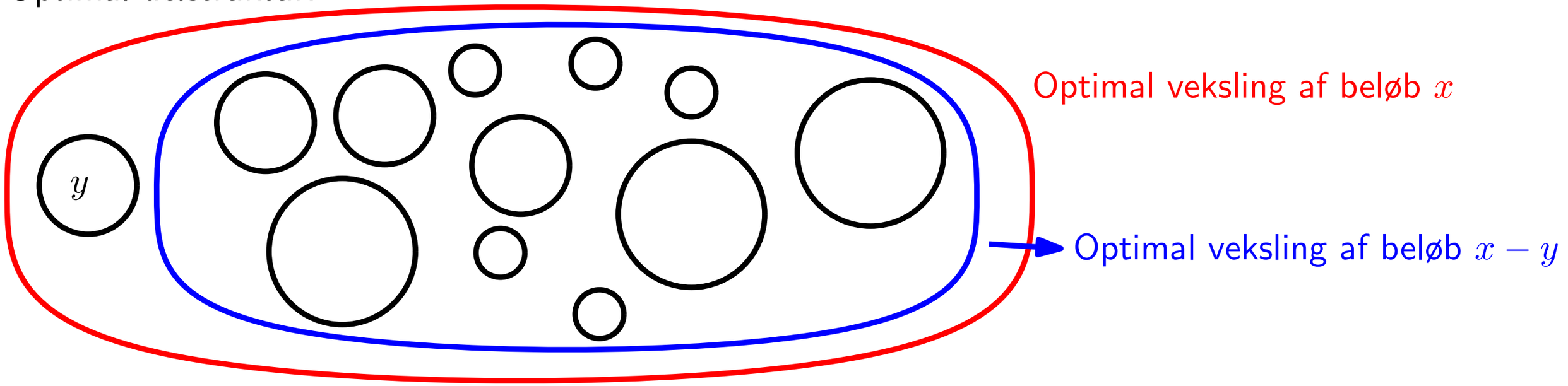
Optimal delstruktur:



Optimal veksling af beløb x

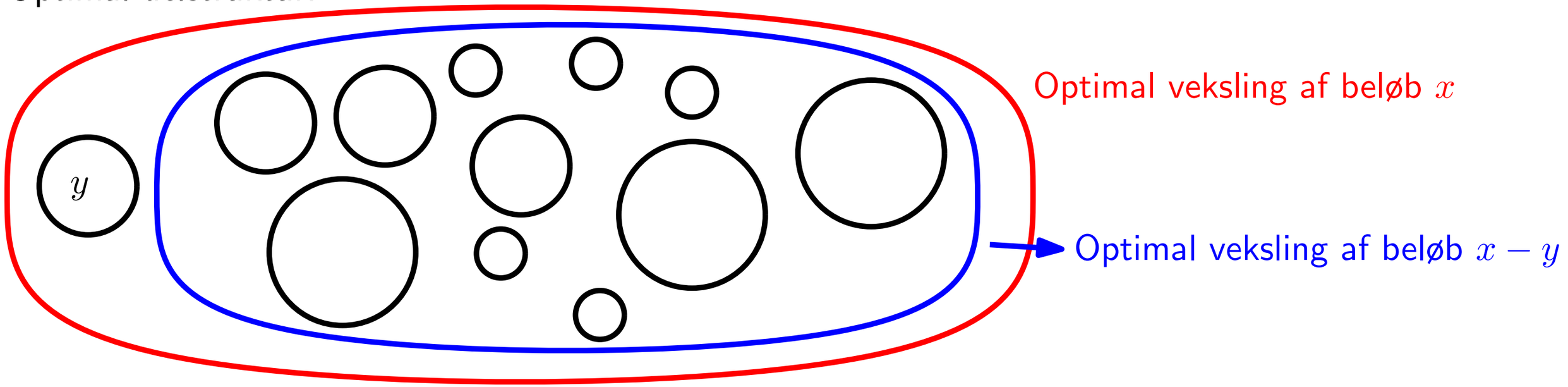
Veksling af mønter

Optimal delstruktur:



Veksling af mønter

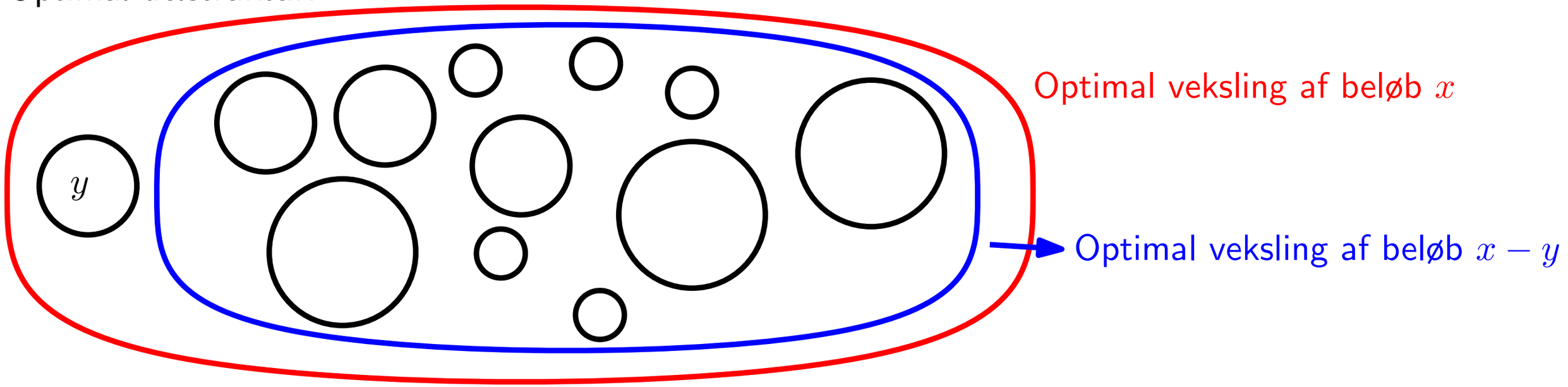
Optimal delstruktur:



Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Veksling af mønter

Optimal delstruktur:



Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

Induktionsbevis

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

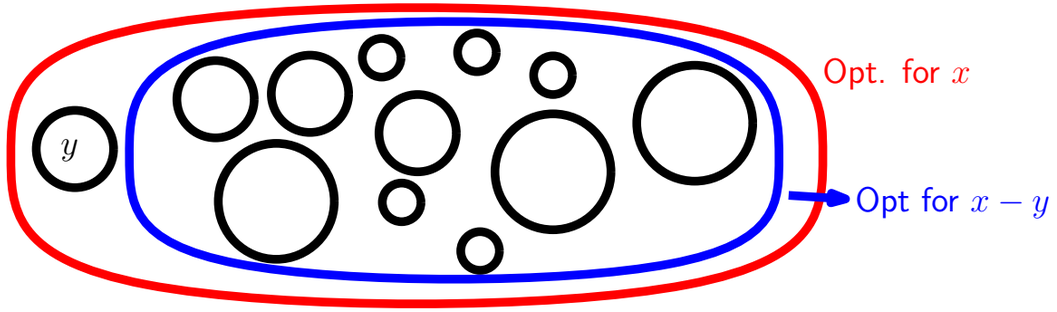
Induktionsbevis over optimalt antal mønter m:

Basistilfælde: $m = 0$. Så $x = 0$. Trivielt.

Induktionshypotese (IH): Algoritmen virker hvis m mønter er nok.

Induktionsskridt: Lad x være beløb hvor optimal løsning bruger $m + 1$ mønter.

Optimal delstruktur:



Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Induktionsbevis

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

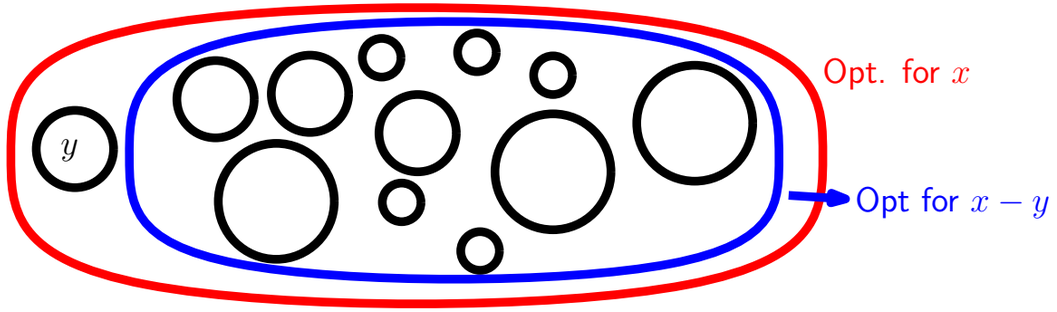
Induktionsbevis over optimalt antal mønter m:

Basistilfælde: $m = 0$. Så $x = 0$. Trivielt.

Induktionshypotese (IH): Algoritmen virker hvis m mønter er nok.

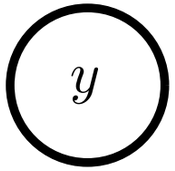
Induktionsskridt: Lad x være beløb hvor optimal løsning bruger $m + 1$ mønter.

Optimal delstruktur:



Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Algoritmen



Optimal løsning

Induktionsbevis

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

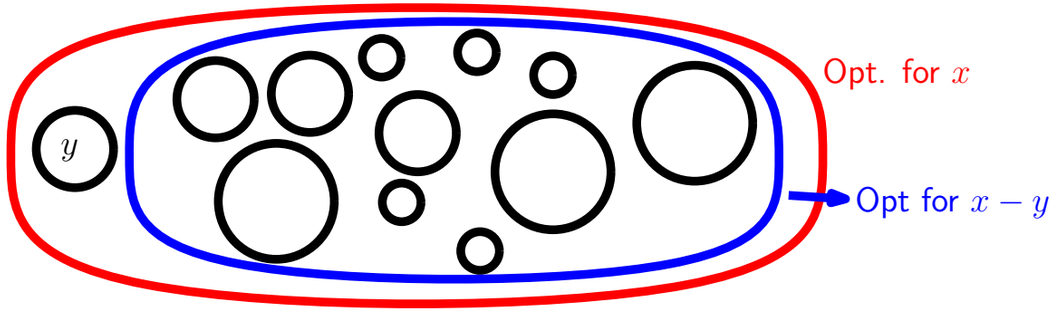
Induktionsbevis over optimalt antal mønter m:

Basistilfælde: $m = 0$. Så $x = 0$. Trivielt.

Induktionshypotese (IH): Algoritmen virker hvis m mønter er nok.

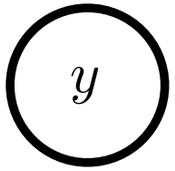
Induktionsskridt: Lad x være beløb hvor optimal løsning bruger $m + 1$ mønter.

Optimal delstruktur:



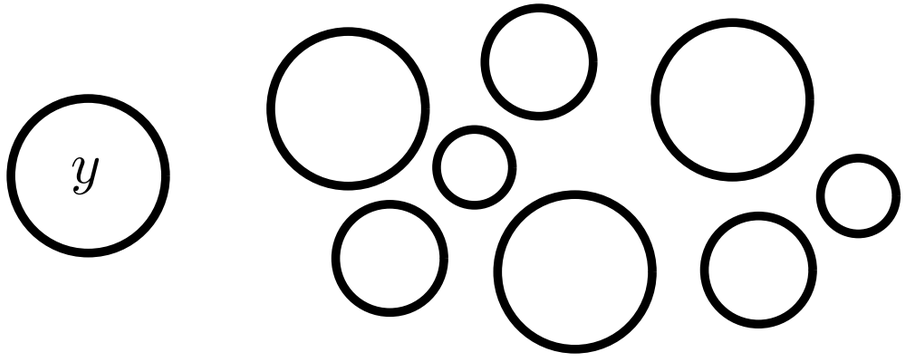
Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Algoritmen



Greedy choice property $\Rightarrow$

Optimal løsning



Induktionsbevis

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

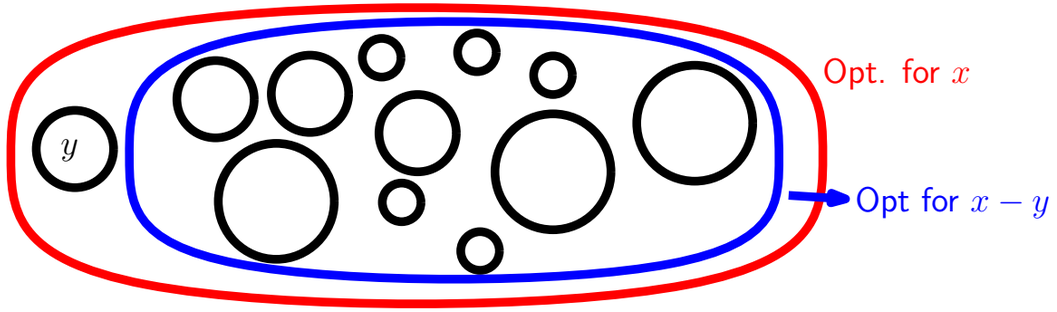
Induktionsbevis over optimalt antal mønter m:

Basistilfælde: $m = 0$. Så $x = 0$. Trivielt.

Induktionshypotese (IH): Algoritmen virker hvis m mønter er nok.

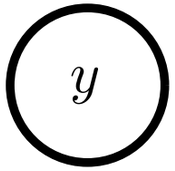
Induktionsskridt: Lad x være beløb hvor optimal løsning bruger $m + 1$ mønter.

Optimal delstruktur:



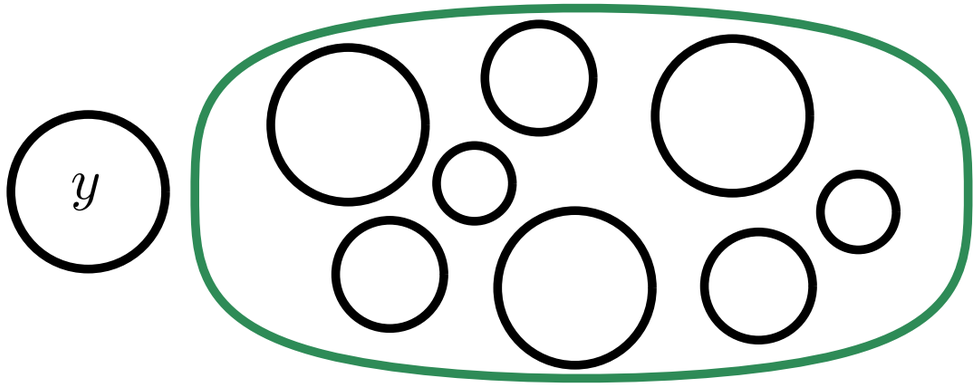
Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

Algoritmen



Optimal løsning

Greedy choice property $\Rightarrow$



Optimal delstruktur $\Rightarrow$ m mønter er opt. for $x - y$

Induktionsbevis

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

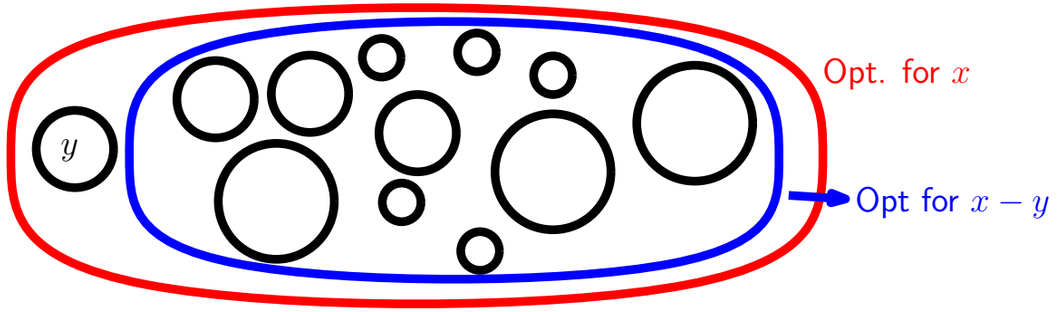
Induktionsbevis over optimalt antal mønter m:

Basistilfælde: $m = 0$. Så $x = 0$. Trivielt.

Induktionshypotese (IH): Algoritmen virker hvis m mønter er nok.

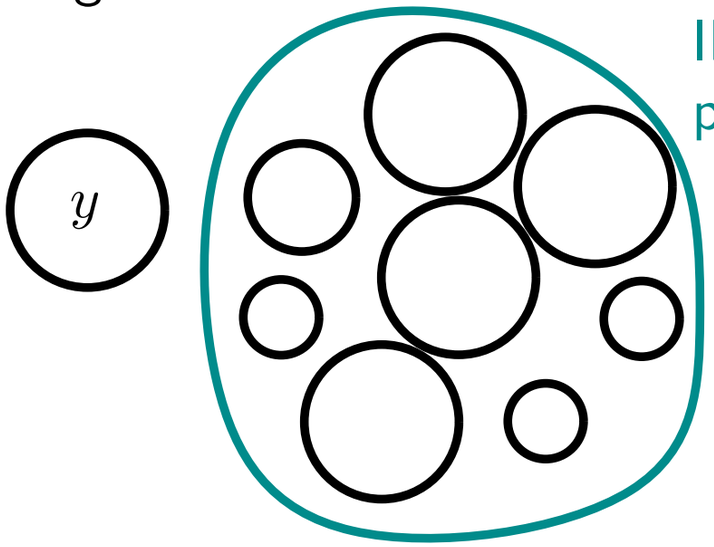
Induktionsskridt: Lad x være beløb hvor optimal løsning bruger $m + 1$ mønter.

Optimal delstruktur:



Greedy choice property: For ethvert beløb x vælger algoritmen en mønt som er med i en optimal løsning.

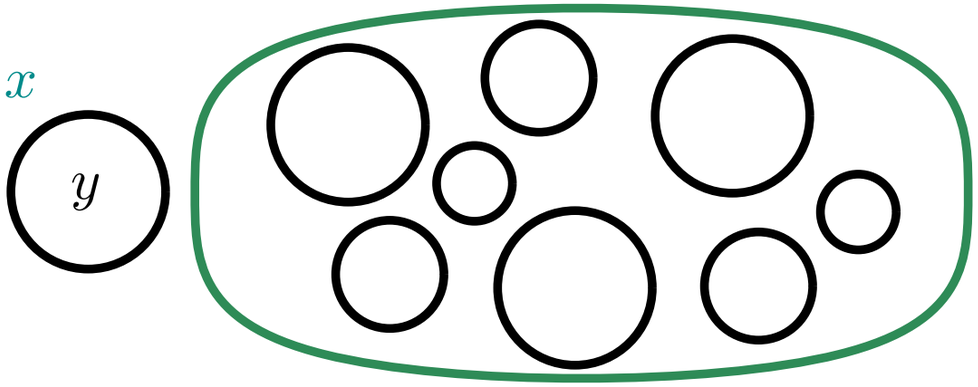
Algoritmen



IH $\Rightarrow$ alg. bruger m mønter på $x - y$ og altså $m + 1$ på x

Greedy choice property $\Rightarrow$

Optimal løsning



Optimal delstruktur $\Rightarrow$ m mønter er opt. for $x - y$

Greedy choice property

Beløb x.

	Grådig algoritme	Optimal løsning
Tilfælde 1: $x = 1$.	①	①
Tilfælde 2: $2 \leq x \leq 4$.	② ⋯	② ⋯
Tilfælde 3: $x \geq 5$.	⑤ ⋯	⑤ ⋯ eller ~~② ② ② ⋯~~ eller ~~② ② ① ⋯~~ eller ~~② ① ① ① ⋯~~ eller ~~① ① ① ① ① ⋯~~

Optimal delstruktur + greedy choice property $\Rightarrow$ algoritmen finder en optimal løsning for alle beløb x

Problem uden greedy choice property

$x = 6$:

Grådig algoritme

Optimal løsning

Problem uden optimal delstruktur

Problem: Givet beløb x, find minimum antal mønter hvis sum er x, med højst én 5'er.
Find a_5, a_2, a_1 så $5a_5 + 2a_2 + a_1 = x$ og $a_5 \leq 1$ og minimér $a_5 + a_2 + a_1$.

Eksempel: $x = 11$
Optimal løsning

Problem uden optimal delstruktur

Problem: Givet beløb x, find minimum antal mønter hvis sum er x, med højst én 5'er.
Find a_5, a_2, a_1 så $5a_5 + 2a_2 + a_1 = x$ og $a_5 \leq 1$ og minimér $a_5 + a_2 + a_1$.

Eksempel: $x = 11$

Optimal løsning

Ikke optimal løsning til beløb $x = 6$.

Problem uden optimal delstruktur

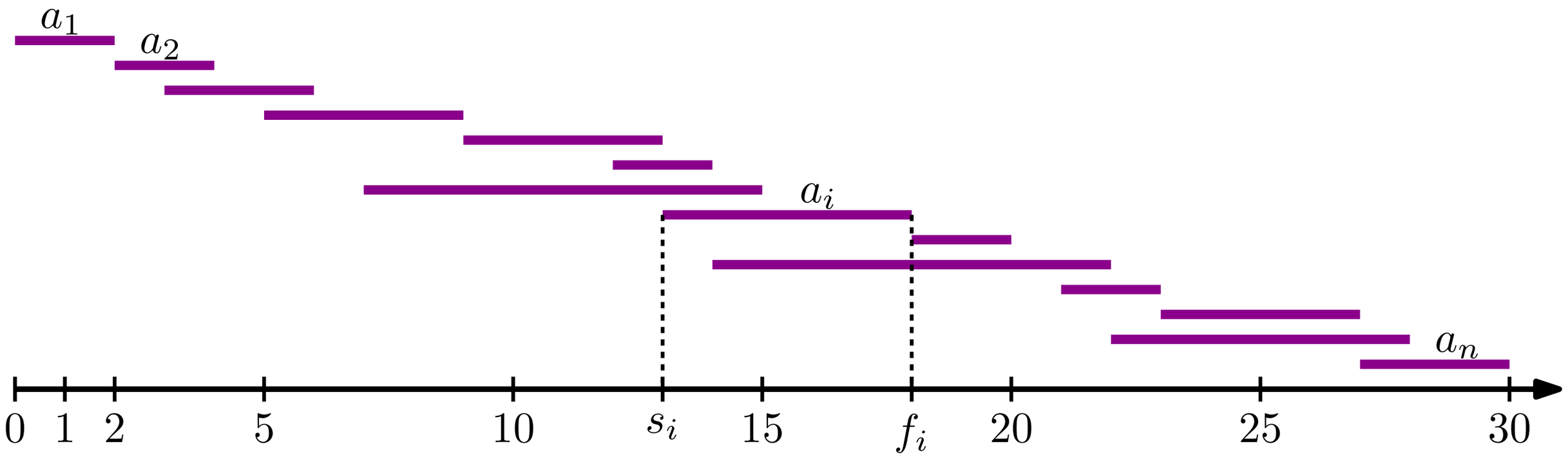
Problem: Givet beløb x, find minimum antal mønter hvis sum er x, med højst én 5'er.
Find a_5, a_2, a_1 så $5a_5 + 2a_2 + a_1 = x$ og $a_5 \leq 1$ og minimér $a_5 + a_2 + a_1$.

Eksempel: $x = 11$
Optimal løsning

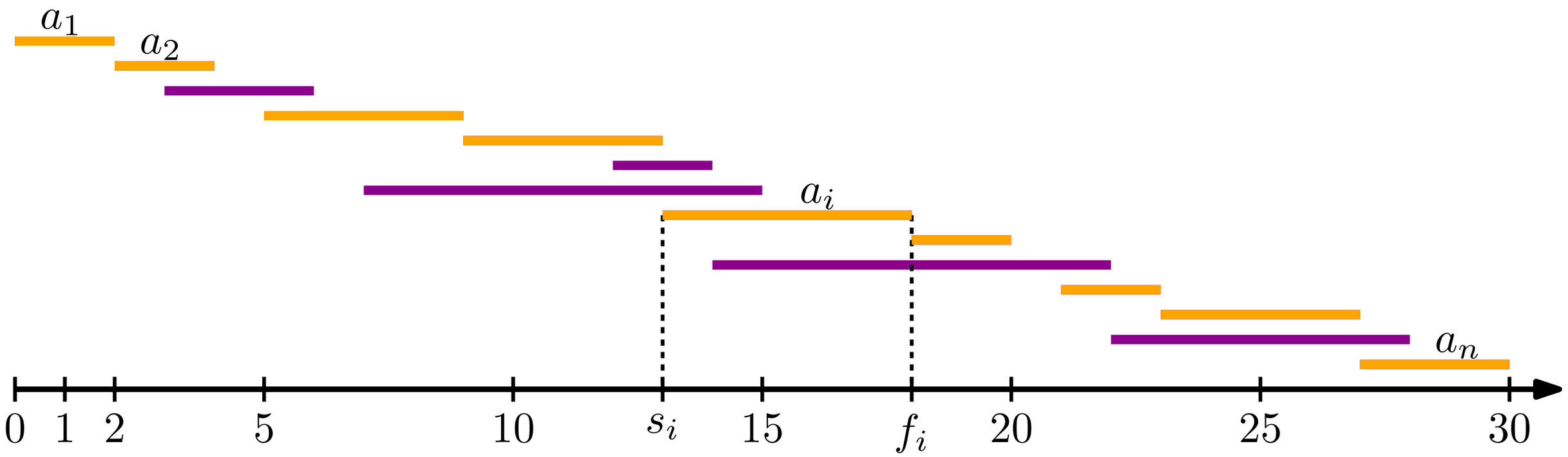
Ikke optimal løsning til beløb $x = 6$.

Greedy choice property uden optimal delstruktur $\Rightarrow$ algoritmen finder ugyldige løsninger.

Aktivitetsudvælgelse (Activity selection)



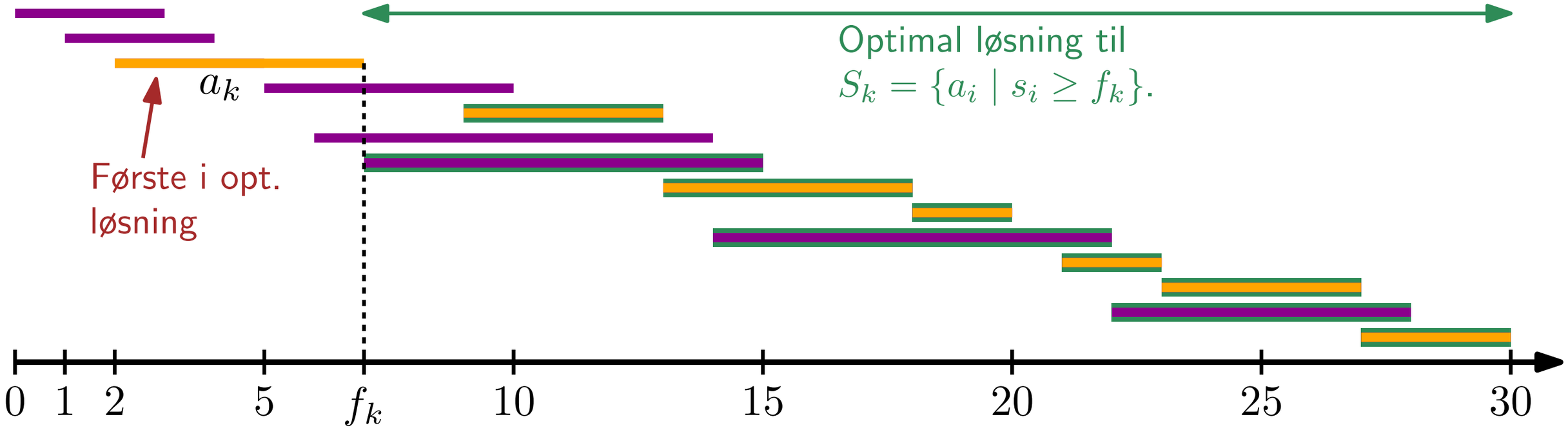
Aktivitetsudvælgelse (Activity selection)



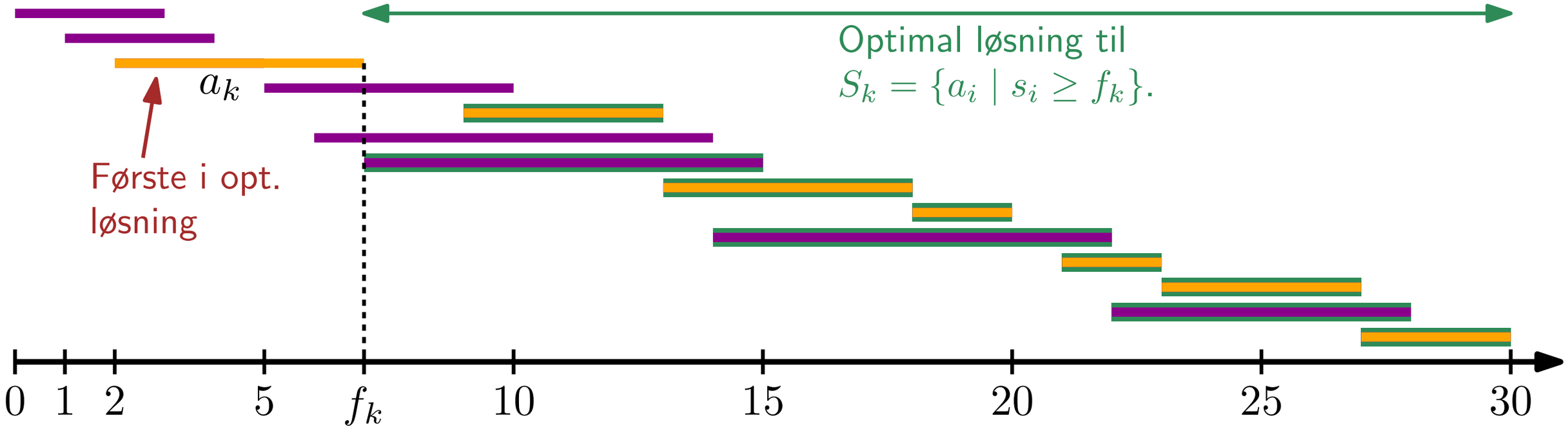
Problem: Find maksimum antal ikke-overlappende aktiviteter.

Antag $f_1 \leq f_2 \leq \cdots \leq f_n$.

Optimal delstruktur



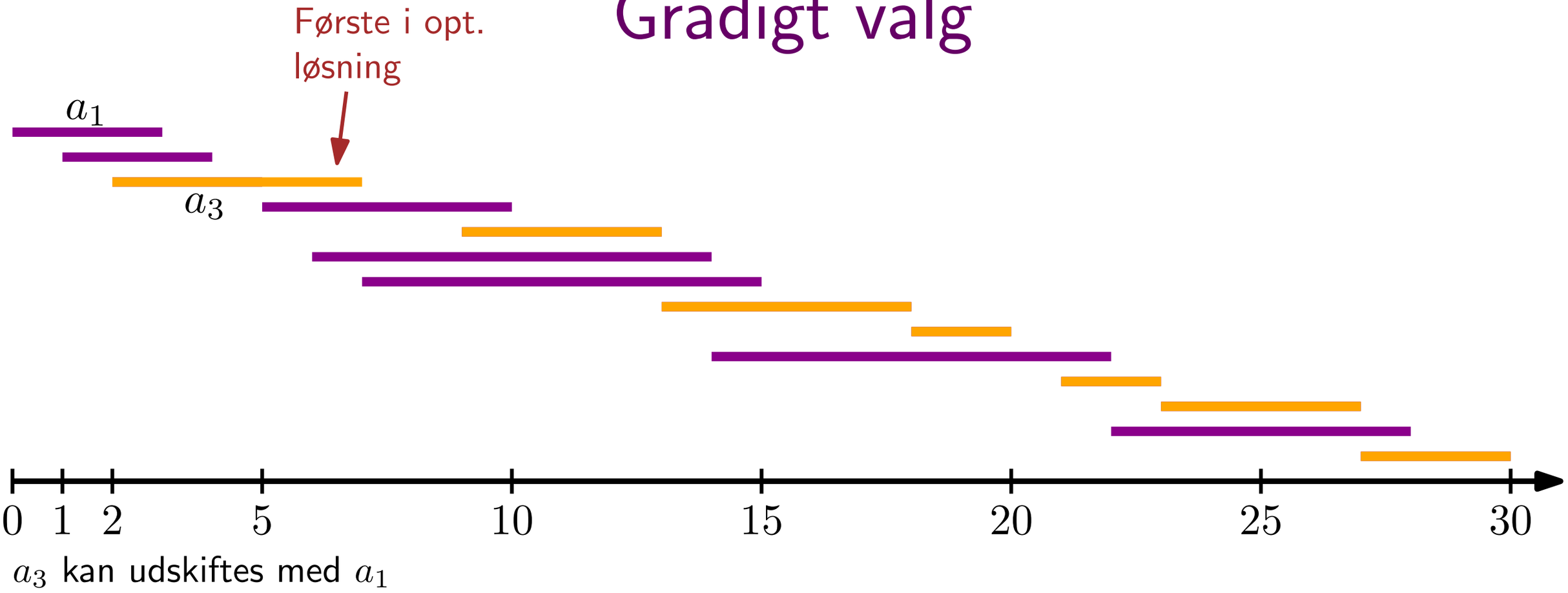
Optimal delstruktur



Dynamisk programmering:
Prøv alle n muligheder for a_k.
Løs S_k rekursivt.
Brug memoisering.

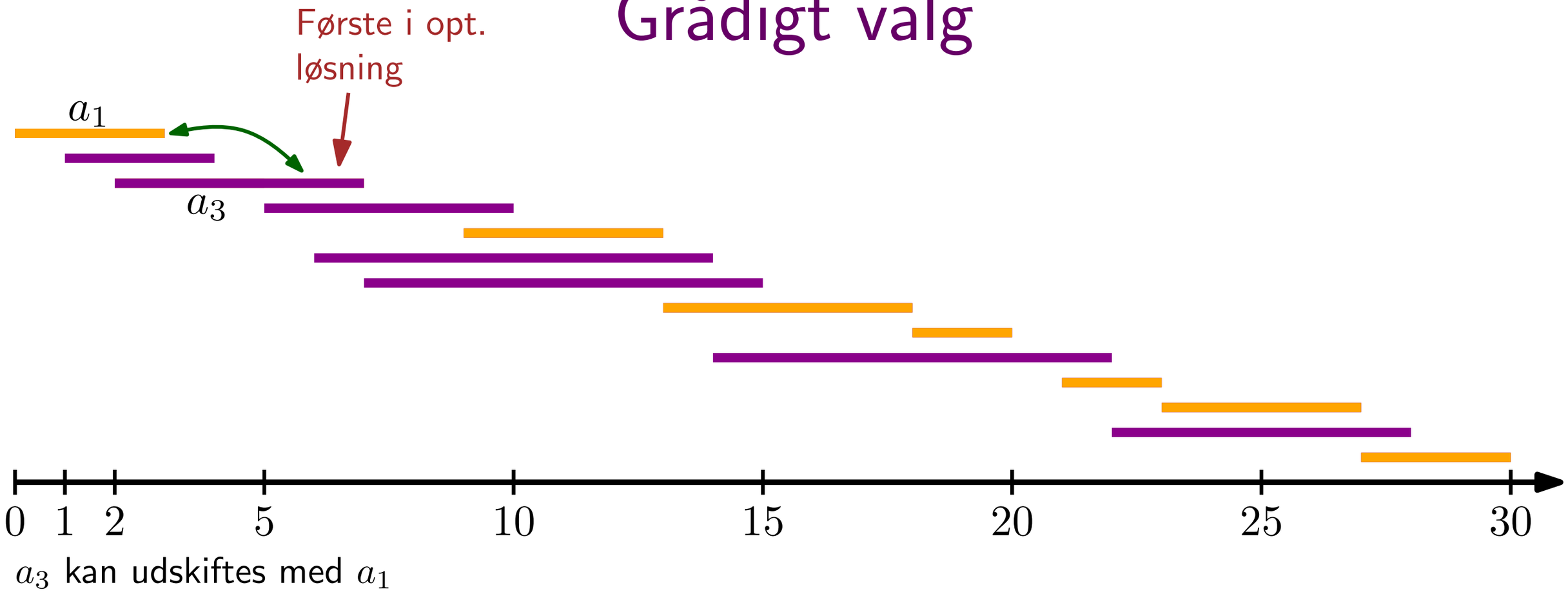
Resultat: n delproblemer S_k. Hvert tager $O(n)$ tid. I alt $O(n^2)$ tid.

Grådigt valg



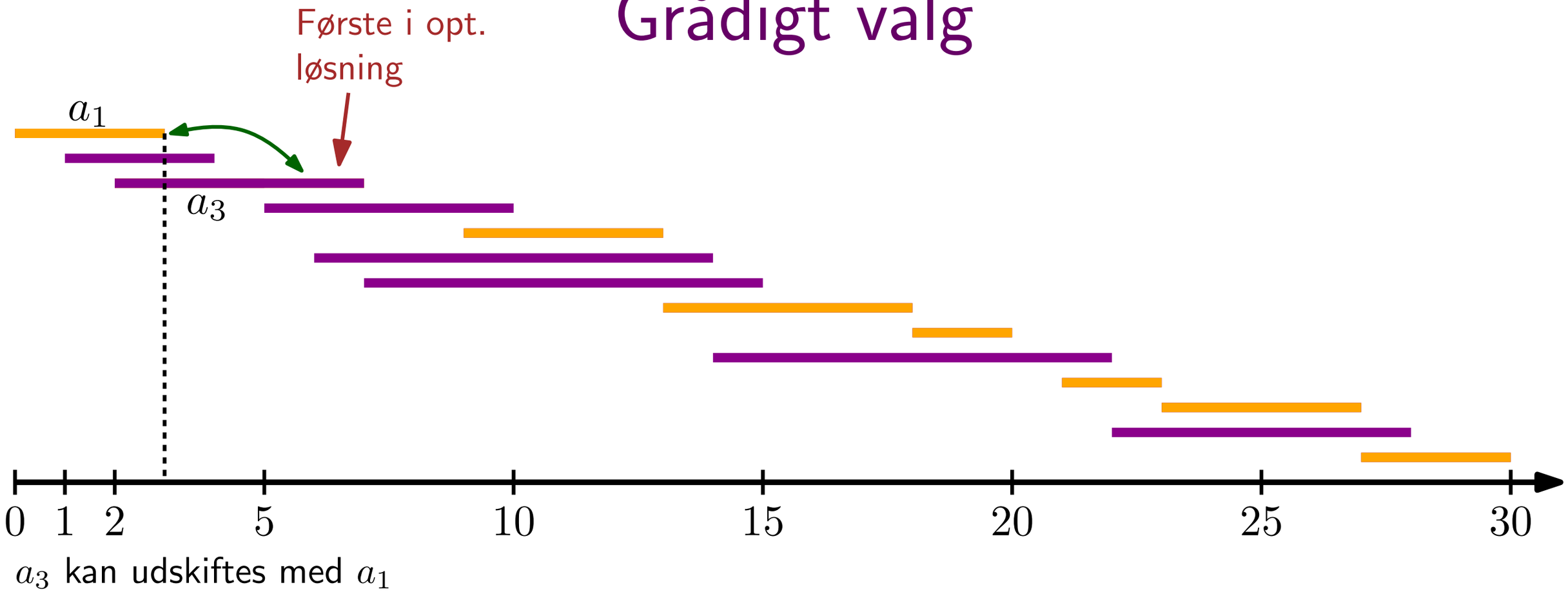
a_3 kan udskiftes med a_1

Grådigt valg



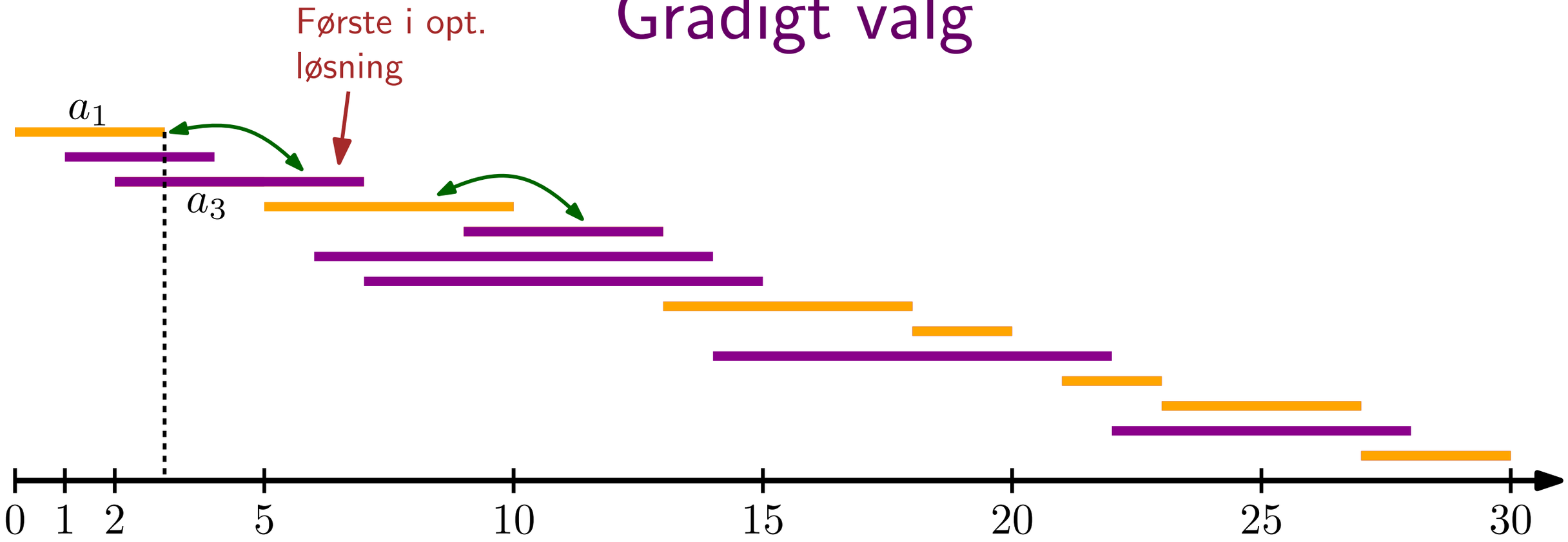
a_3 kan udskiftes med a_1

Grådigt valg



a_3 kan udskiftes med a_1

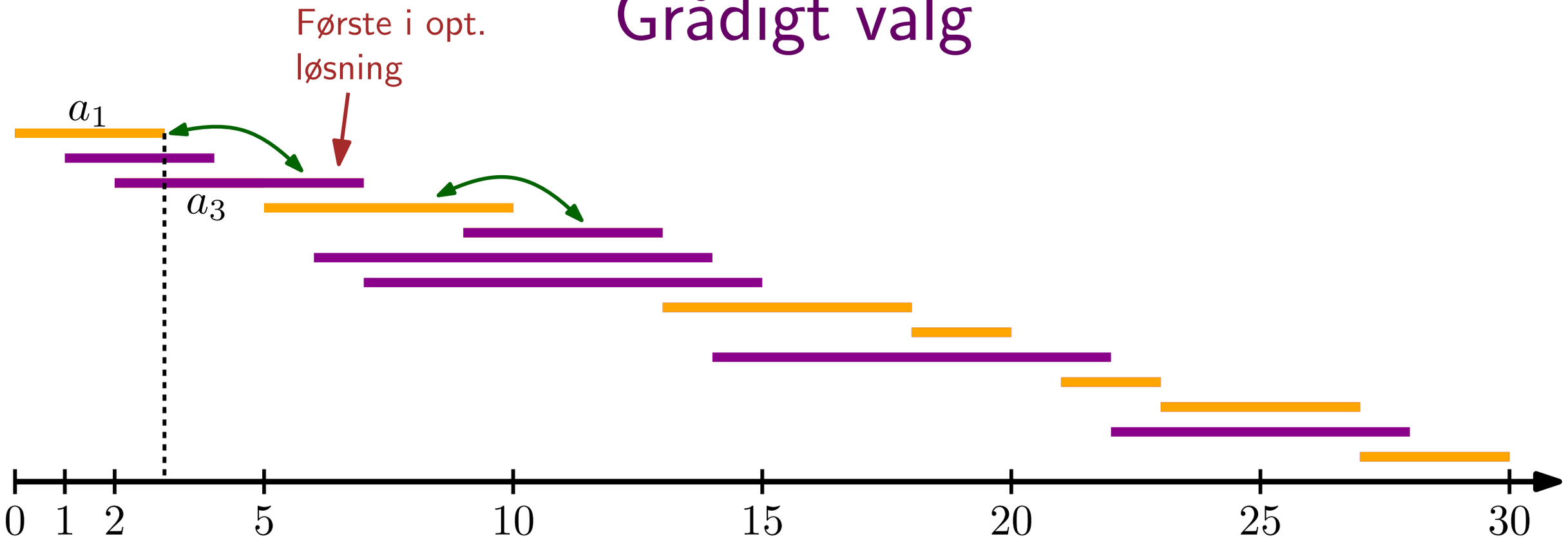
Grådigt valg



a_3 kan udskiftes med a_1

a_5 kan udskiftes med a_4

Grådigt valg

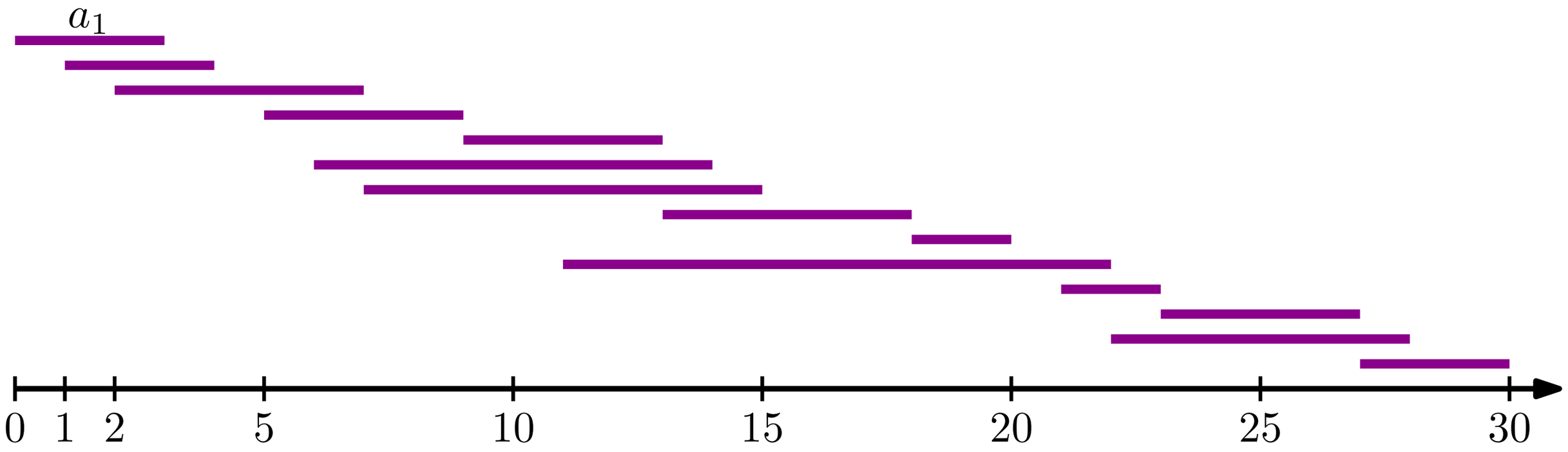


a_3 kan udskiftes med a_1

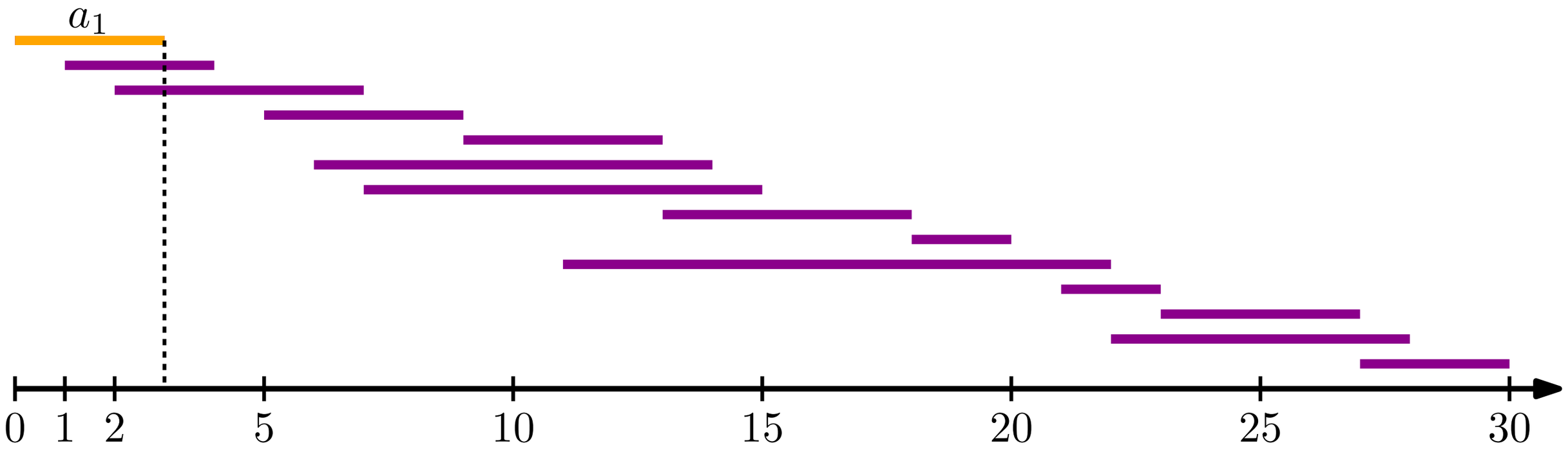
a_5 kan udskiftes med a_4

Grådigt valg: Efter a_k vælges den første aktivitet i S_k.
$S_k = \{a_i \mid s_i \geq f_k\}$

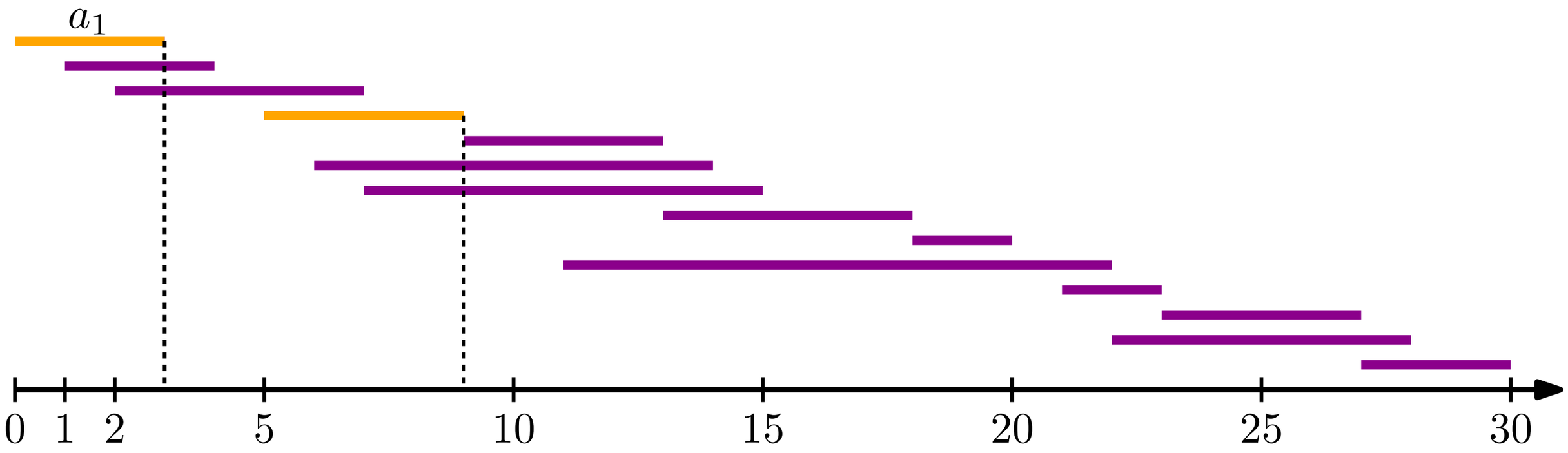
Grådig algoritme



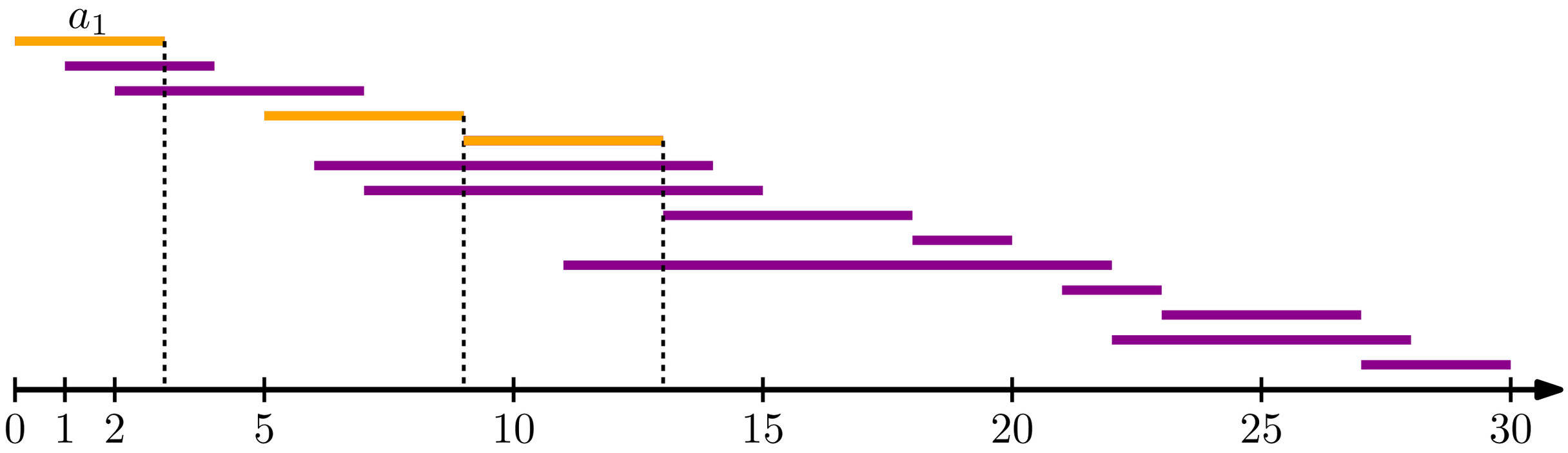
Grådig algoritme



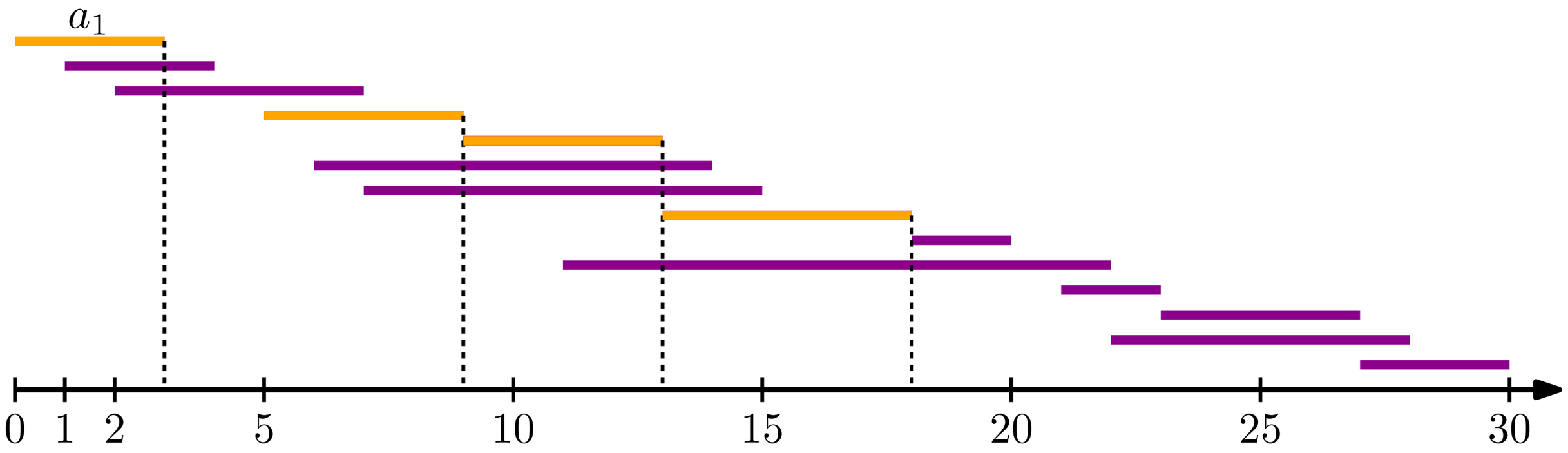
Grådig algoritme



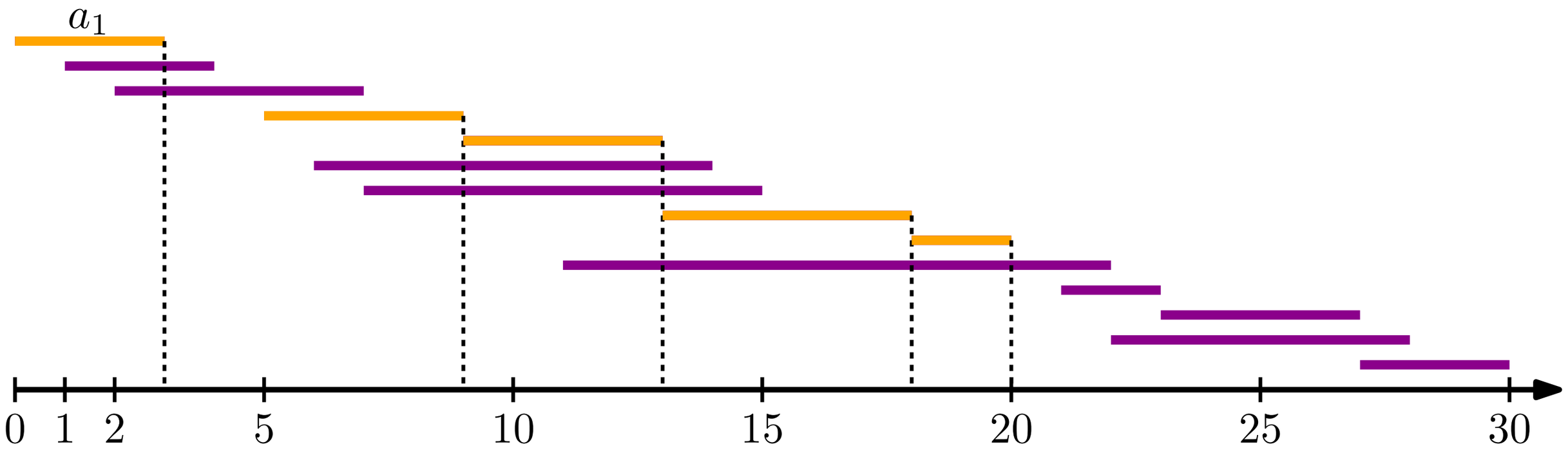
Grådig algoritme



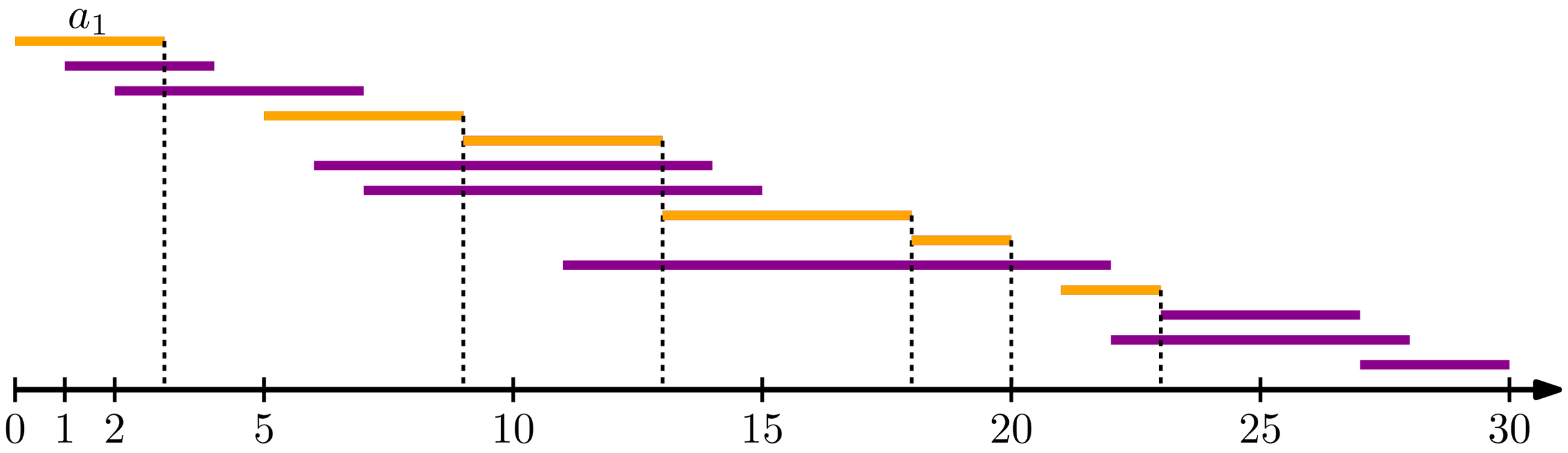
Grådig algoritme



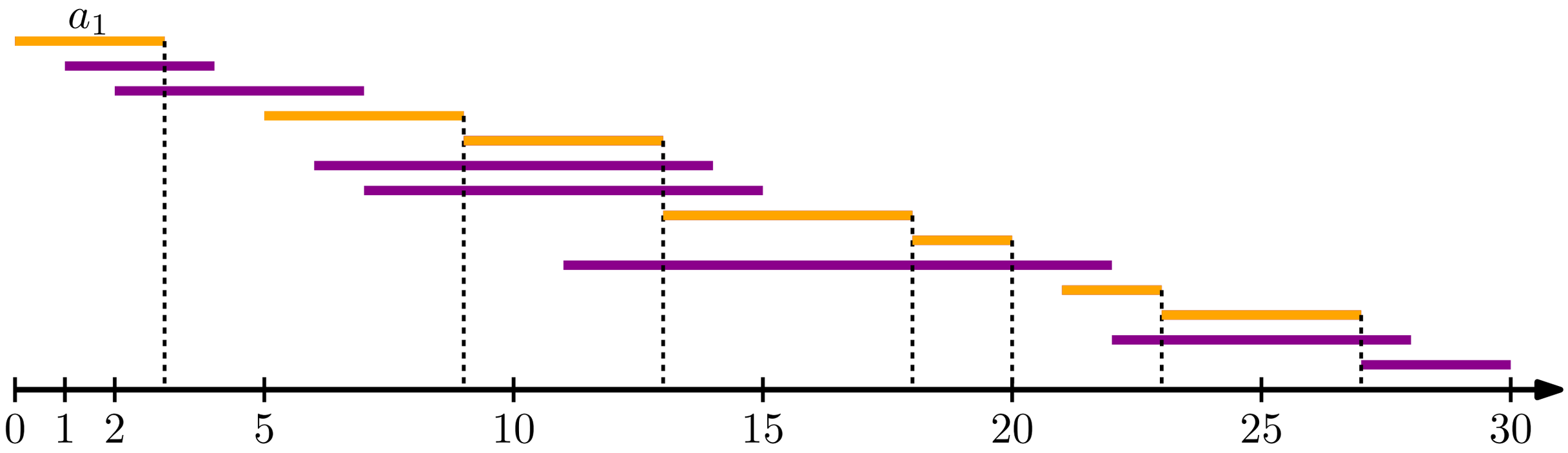
Grådig algoritme



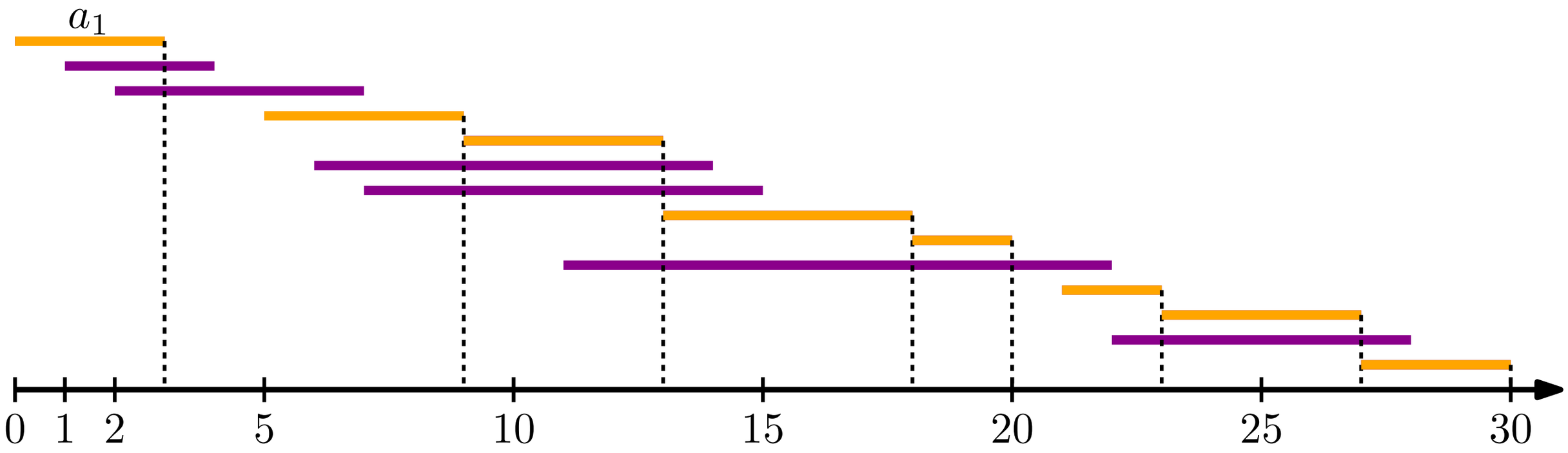
Grådig algoritme



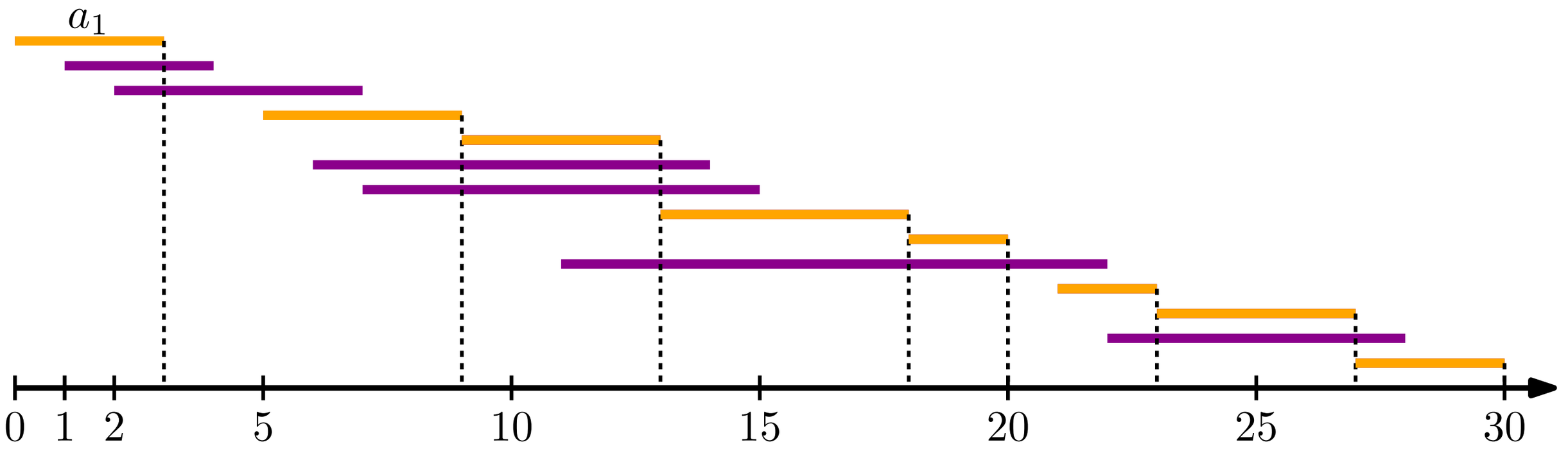
Grådig algoritme



Grådig algoritme



Grådig algoritme



GREEDY-ACTIVITY-SELECTOR(s, f, n)

```
1  A = {a_1}
2  k = 1
3  for m = 2 to n
4      if s[m] ≥ f[k]          // is a_m in S_k?
5          A = A ∪ {a_m}       // yes, so choose it
6          k = m               // and continue from there
7  return A
```

$\Theta(n)$ tid.

Repræsentation af tekst

Tekst T består af bogstaver a, b, c, d, e, f.

Hvordan repræsenterer vi T?

	a	b	c	d	e	f
ASCII	01100001	01100010	01100011	01100100	01100101	01100110

Repræsentation af tekst

Tekst T består af bogstaver a, b, c, d, e, f.
Hvordan repræsenterer vi T?

	a	b	c	d	e	f
ASCII	01100001	01100010	01100011	01100100	01100101	01100110

ASCII: badeabe $=$ 01100010 01100001 01100100 01100101 01100001 01100010 01100101

Repræsentation af tekst

Tekst T består af bogstaver a, b, c, d, e, f.
Hvordan repræsenterer vi T?

	a	b	c	d	e	f
ASCII	01100001	01100010	01100011	01100100	01100101	01100110
3 bits	001	010	011	100	101	110

ASCII: badeabe $=$ 01100010 01100001 01100100 01100101 01100001 01100010 01100101

Repræsentation af tekst

Tekst T består af bogstaver a, b, c, d, e, f.
Hvordan repræsenterer vi T?

	a	b	c	d	e	f
ASCII	01100001	01100010	01100011	01100100	01100101	01100110
3 bits	001	010	011	100	101	110

ASCII: badeabe = 01100010 01100001 01100100 01100101 01100001 01100010 01100101

3 bits: badeabe = 010 001 100 101 001 010 101

Repræsentation af tekst

Tekst T består af bogstaver a, b, c, d, e, f.
Hvordan repræsenterer vi T?

	a	b	c	d	e	f
ASCII	01100001	01100010	01100011	01100100	01100101	01100110
3 bits	001	010	011	100	101	110

ASCII: badeabe = 01100010 01100001 01100100 01100101 01100001 01100010 01100101

3 bits: badeabe = 010 001 100 101 001 010 101

Idé: Hvis nogle bogstaver er mere almindelige end andre
i T, kan vi så lave en bedre kode som sparer bits?

Datakompression

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde	0	00	01	1	11	110

Datakompression

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde	0	00	01	1	11	110

Længde af tekst med 3 bit-koder:
$(45 + 13 + 12 + 16 + 9 + 5) \cdot 1000 \cdot 3 = 300000.$

Længde af tekst med koder af variable længde:
$(45 \cdot 1 + 13 \cdot 2 + 12 \cdot 2 + 16 \cdot 1 + 9 \cdot 2 + 5 \cdot 3) \cdot 1000 = 144000.$

Datakompression

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde	0	00	01	1	11	110

Længde af tekst med 3 bit-koder:
$(45 + 13 + 12 + 16 + 9 + 5) \cdot 1000 \cdot 3 = 300000.$

Længde af tekst med koder af variable længde:
$(45 \cdot 1 + 13 \cdot 2 + 12 \cdot 2 + 16 \cdot 1 + 9 \cdot 2 + 5 \cdot 3) \cdot 1000 = 144000.$

Hvad er problemet?

Datakompression

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde præfiksfri	0	101	100	111	1101	1100

Datakompression

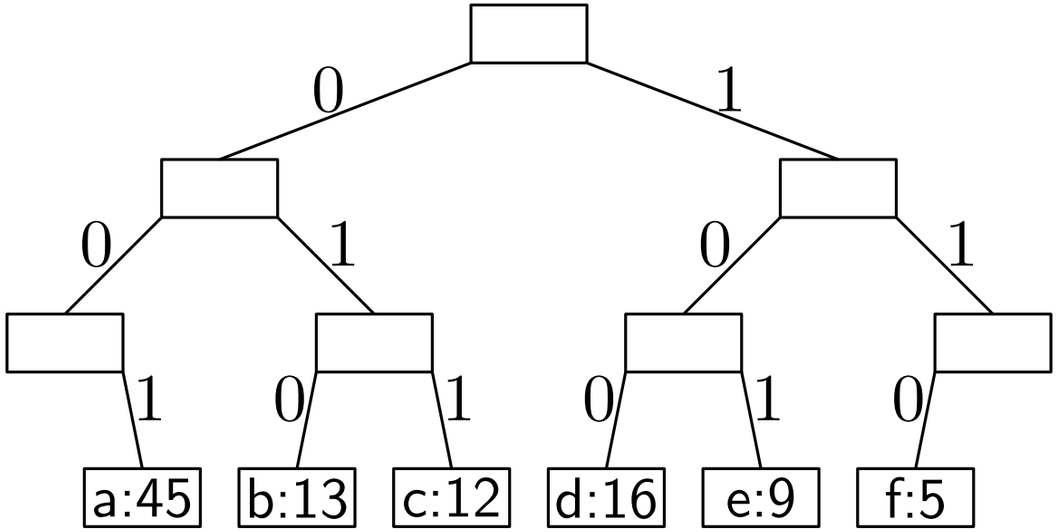
	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde præfiksfri	0	101	100	111	1101	1100

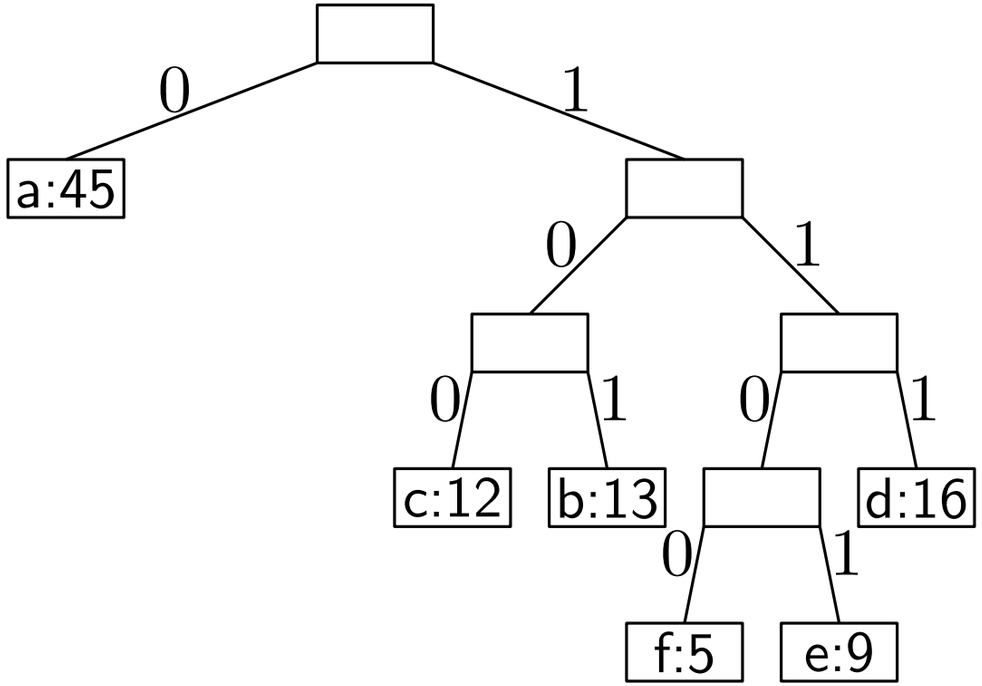
Længde af tekst med 3 bit-koder:
$(45 + 13 + 12 + 16 + 9 + 5) \cdot 1000 \cdot 3 = 300000.$

Længde af tekst med koder af variable længde:
$(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1000 = 224000.$

Parse tree

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde præfiksfri	0	101	100	111	1101	1100

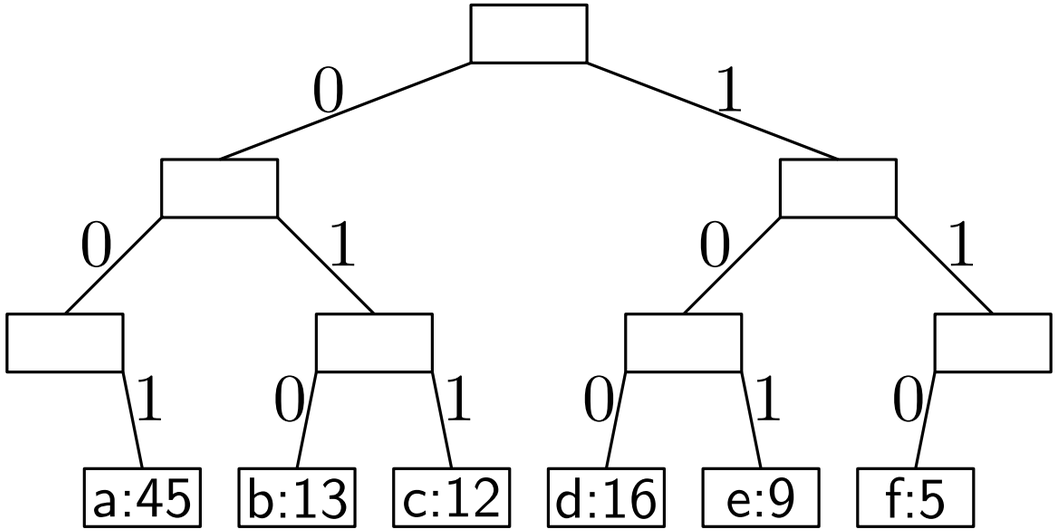


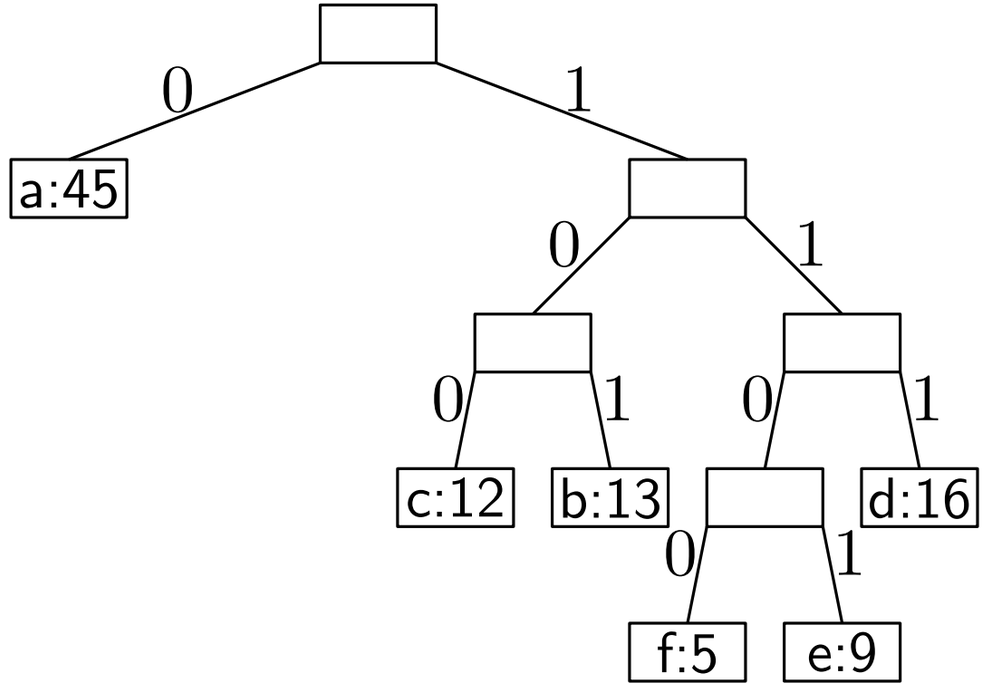


Præfiksfri $\Leftrightarrow$ Alle symboler er i bladene.

Parse tree

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde præfiksfri	0	101	100	111	1101	1100



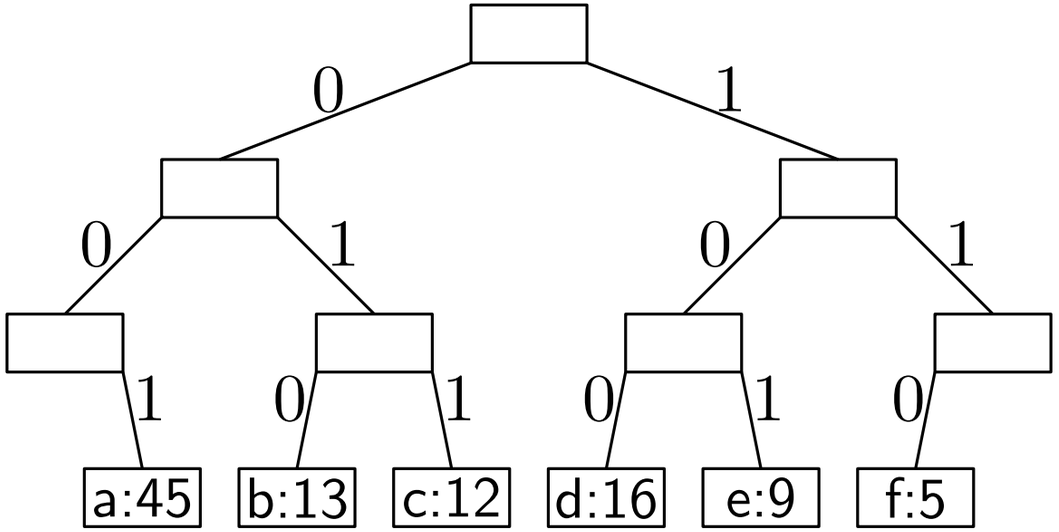


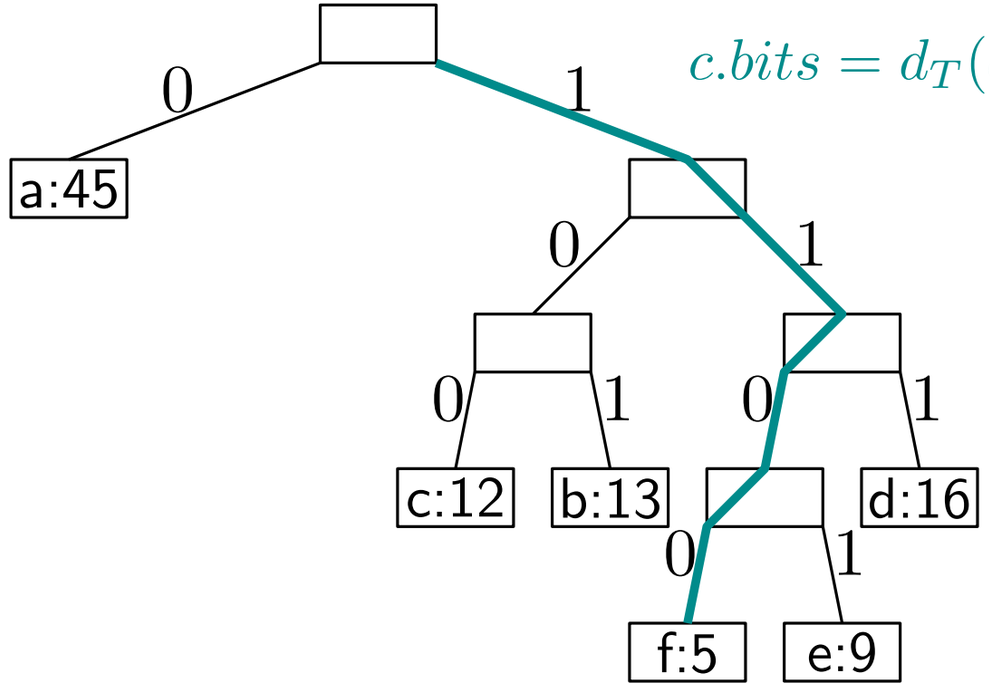
Præfiksfri $\Leftrightarrow$ Alle symboler er i bladene.

$$\#\text{bits} = \sum_{c \in C} c.freq \cdot c.bits$$

Parse tree

	a	b	c	d	e	f
Frekvens/10^3	45	13	12	16	9	5
3 bits	001	010	011	100	101	110
Variabel længde præfiksfri	0	101	100	111	1101	1100





Præfiksfri $\Leftrightarrow$ Alle symboler er i bladene.

$$\#\text{bits} = \sum_{c \in C} c.freq \cdot c.bits$$

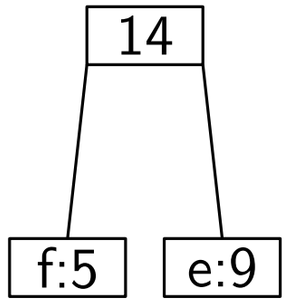
$$B(T) = \sum_{c \in C} c.freq \cdot d_T(c)$$

Huffmans algoritme

a:45 d:16 b:13 c:12 e:9 f:5

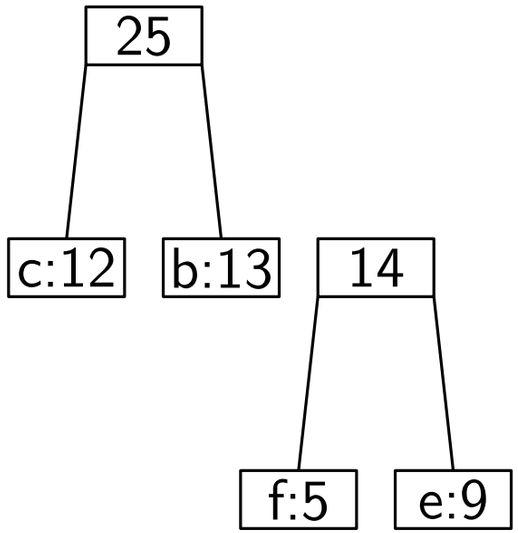
Huffmans algoritme

a:45 d:16 b:13 c:12



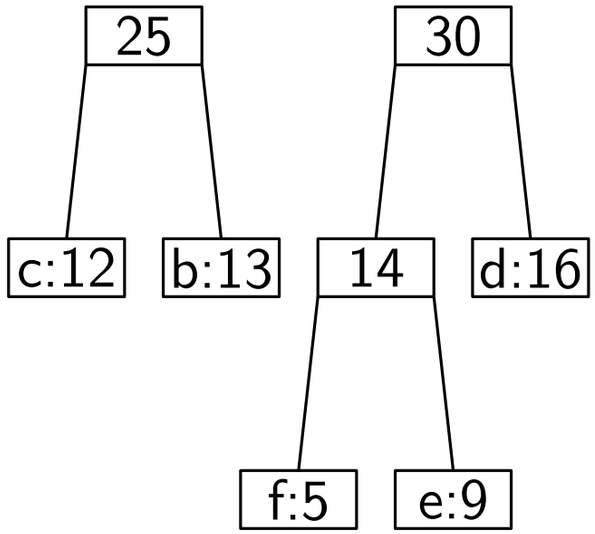
Huffmans algoritme

a:45 d:16



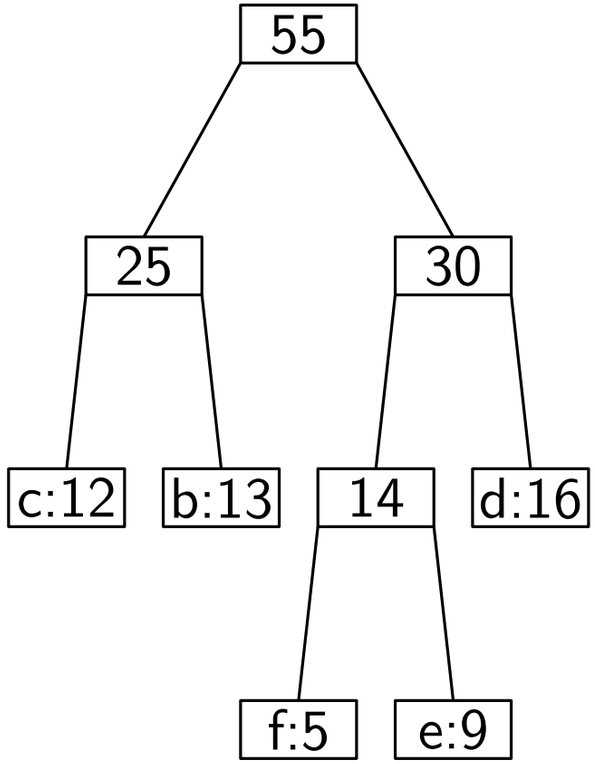
Huffmans algoritme

a:45

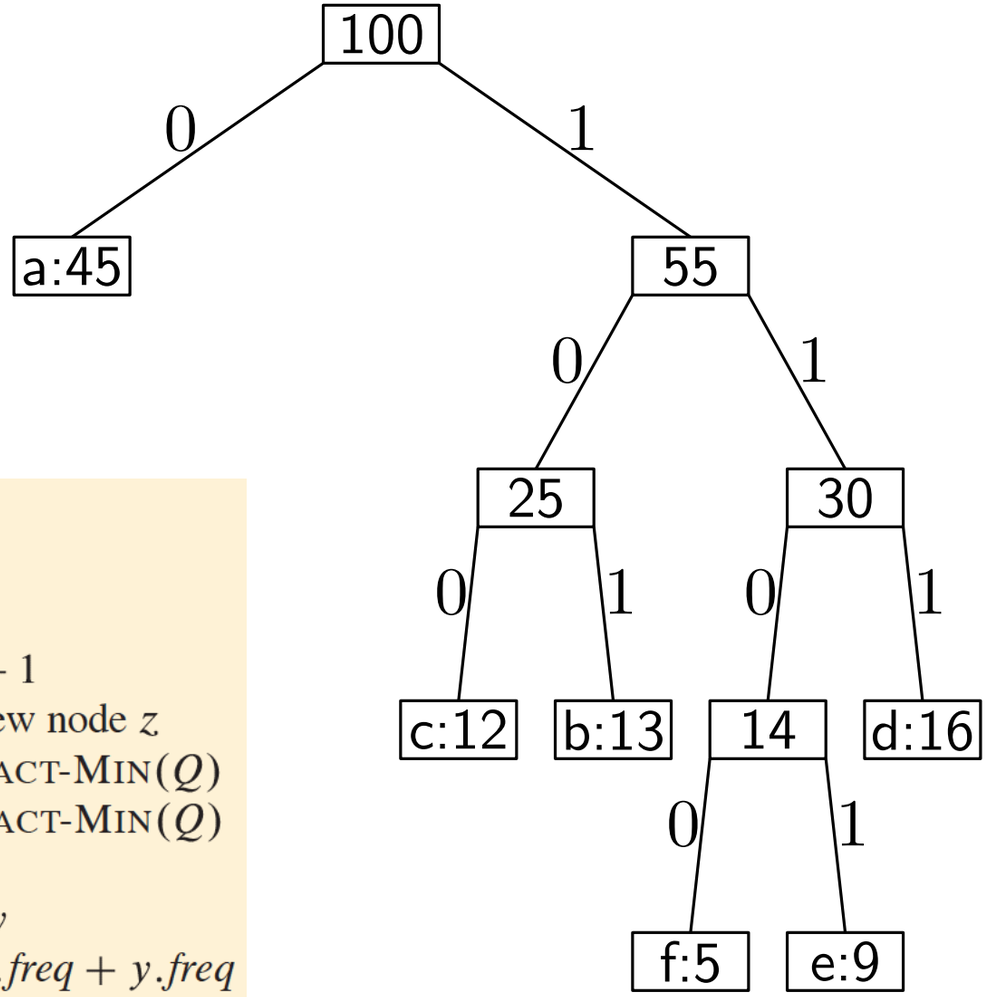


Huffmans algoritme

a:45



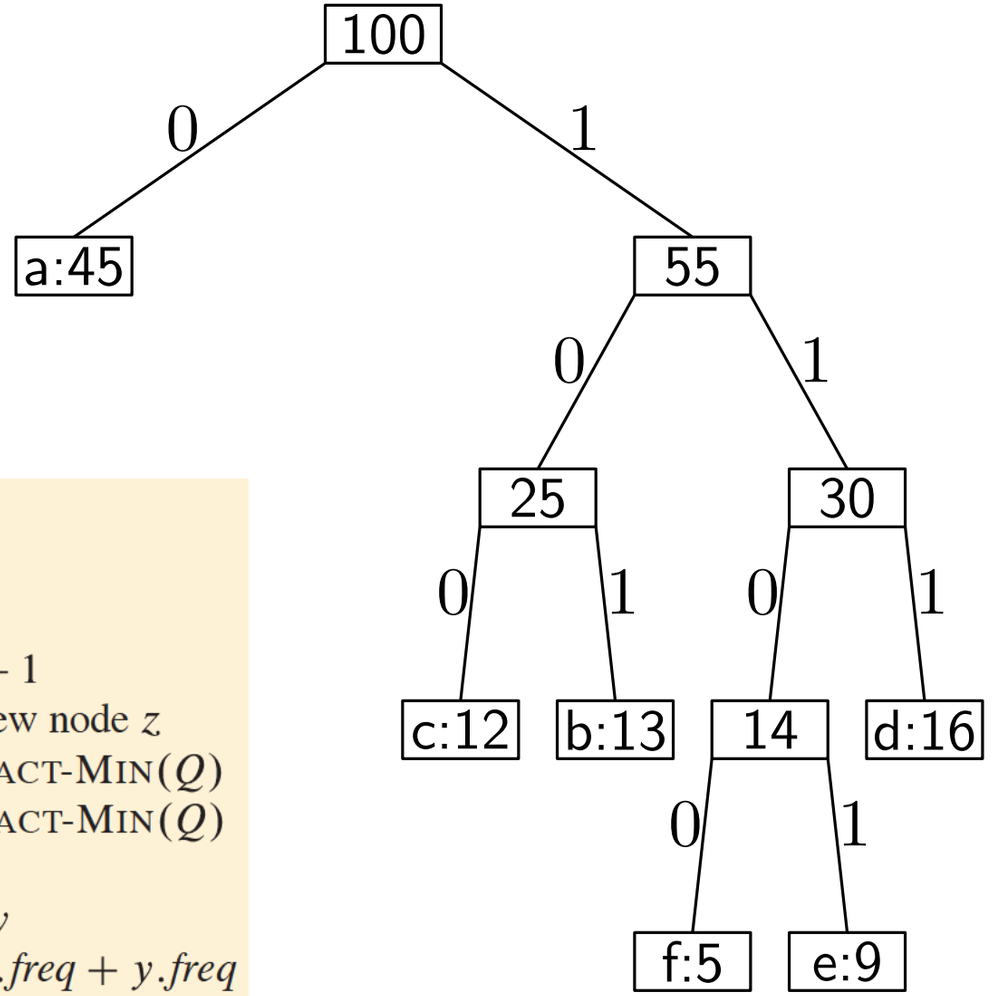
Huffmans algoritme




```
HUFFMAN(C)
1   n = |C|
2   Q = C
3   for i = 1 to n − 1
4       allocate a new node z
5       x = EXTRACT-MIN(Q)
6       y = EXTRACT-MIN(Q)
7       z.left = x
8       z.right = y
9       z.freq = x.freq + y.freq
10      INSERT(Q, z)
11  return EXTRACT-MIN(Q)
```

Huffmans algoritme

Køretid i alt:
$O(n \log n)$

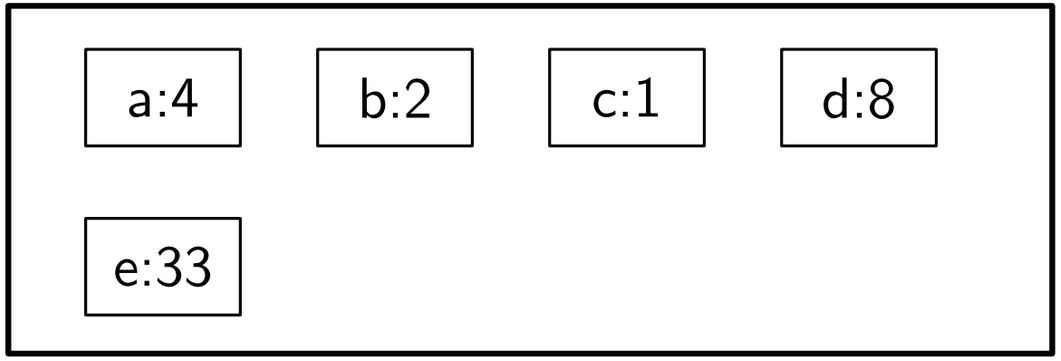


Lave min-hob: $O(n)$

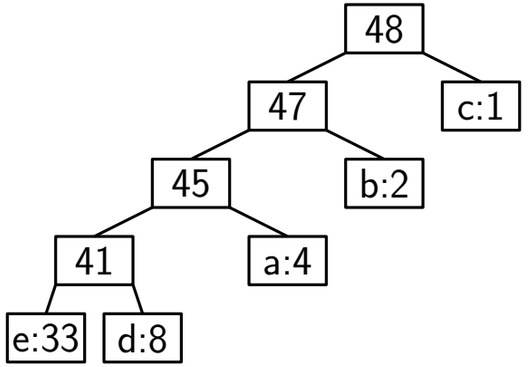
$O(\log n)$

```
HUFFMAN(C)
1   n = |C|
2   Q = C
3   for i = 1 to n − 1
4       allocate a new node z
5       x = EXTRACT-MIN(Q)
6       y = EXTRACT-MIN(Q)
7       z.left = x
8       z.right = y
9       z.freq = x.freq + y.freq
10      INSERT(Q, z)
11  return EXTRACT-MIN(Q)
```

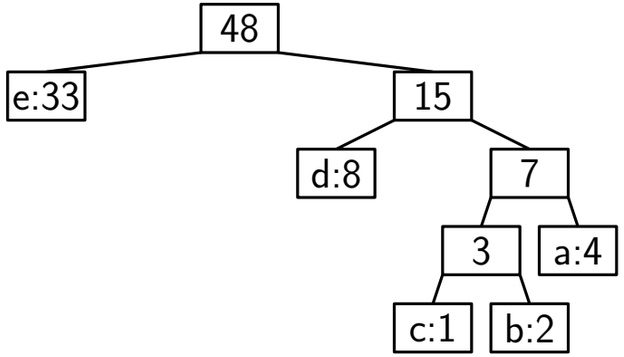
Hvordan bliver parse tree?




```
HUFFMAN(C)
1  n = |C|
2  Q = C
3  for i = 1 to n − 1
4      allocate a new node z
5      x = EXTRACT-MIN(Q)
6      y = EXTRACT-MIN(Q)
7      z.left = x
8      z.right = y
9      z.freq = x.freq + y.freq
10     INSERT(Q, z)
11 return EXTRACT-MIN(Q)
```

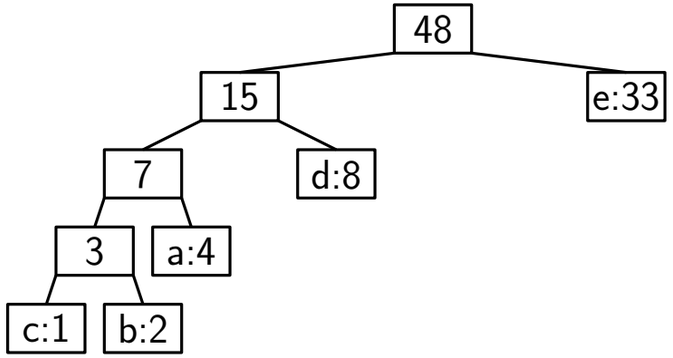


A

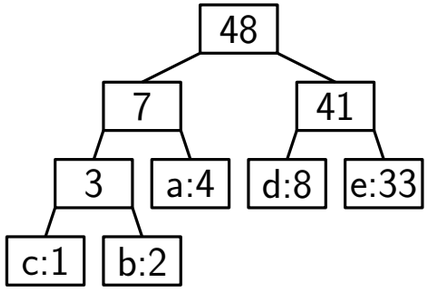


B

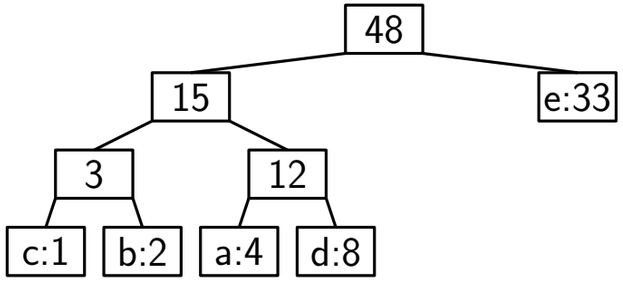
socrative.com → Student login,
Room name: ABRAHAMSEN3464



C



D



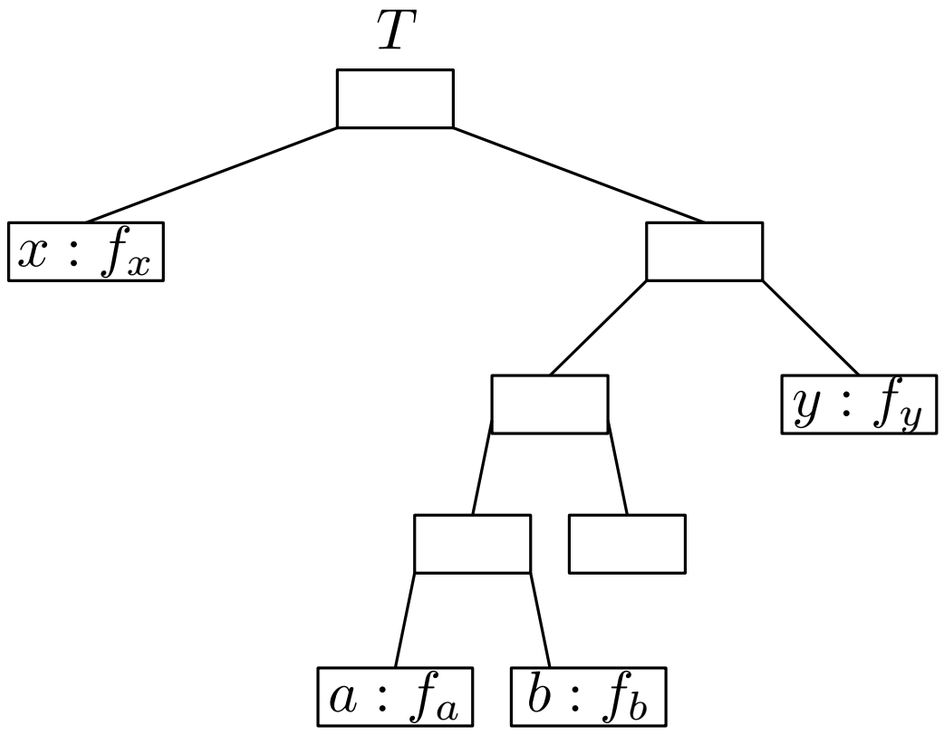
E

Greedy choice property

Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$

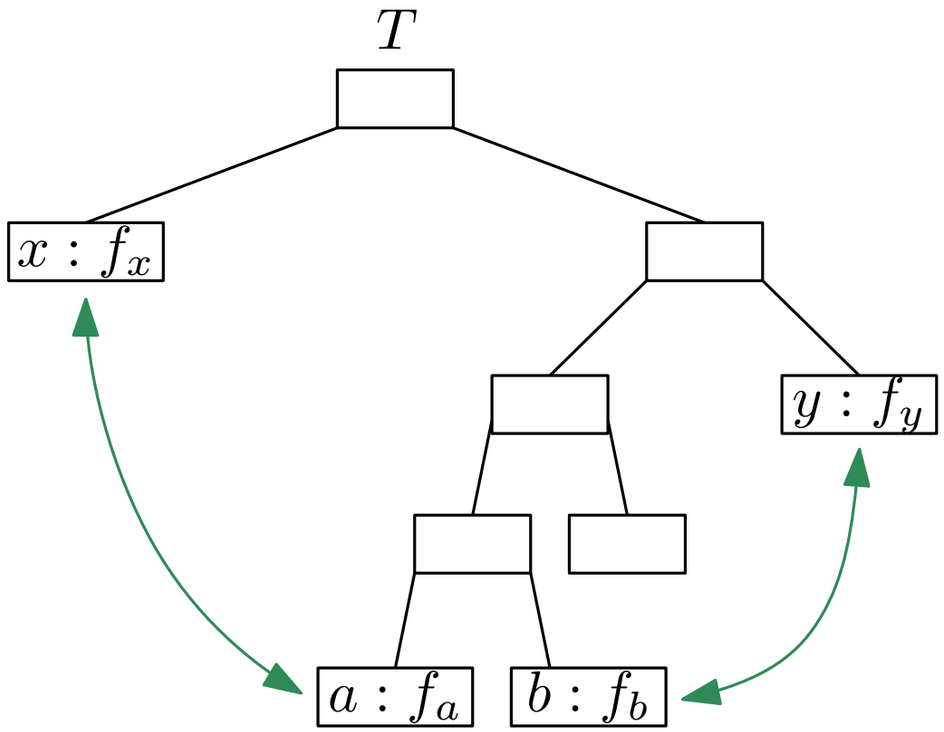


Greedy choice property

Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$

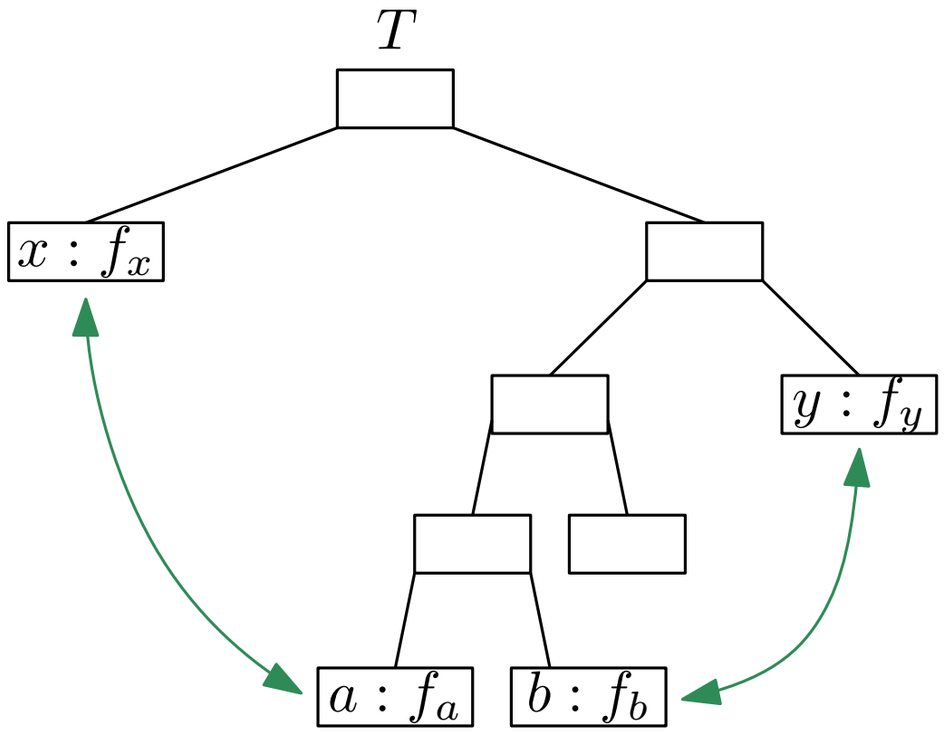


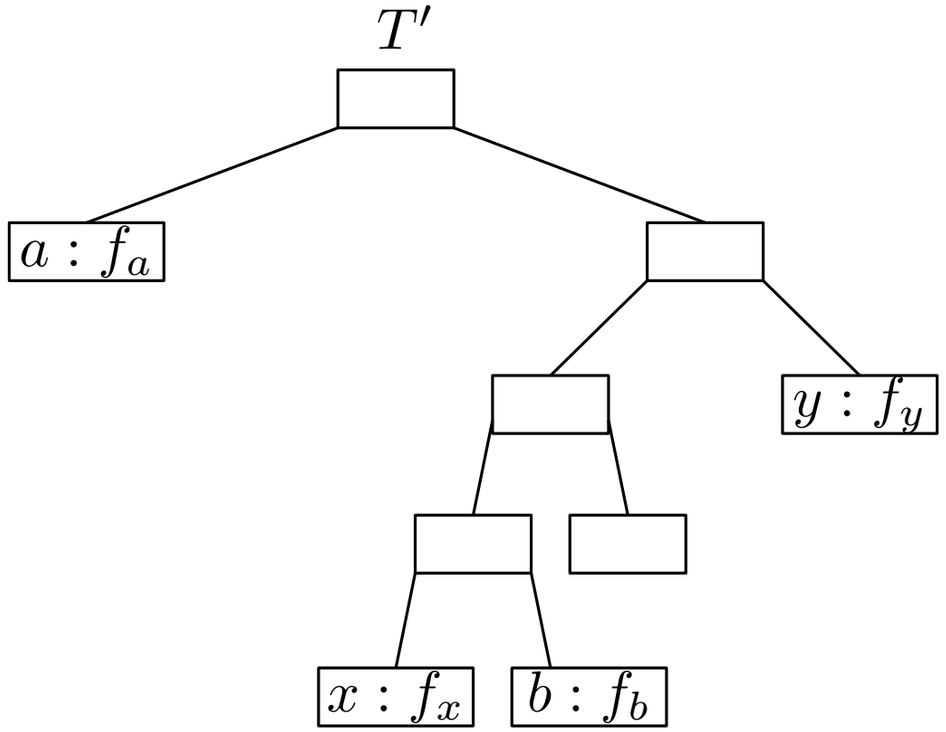
Greedy choice property

Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$





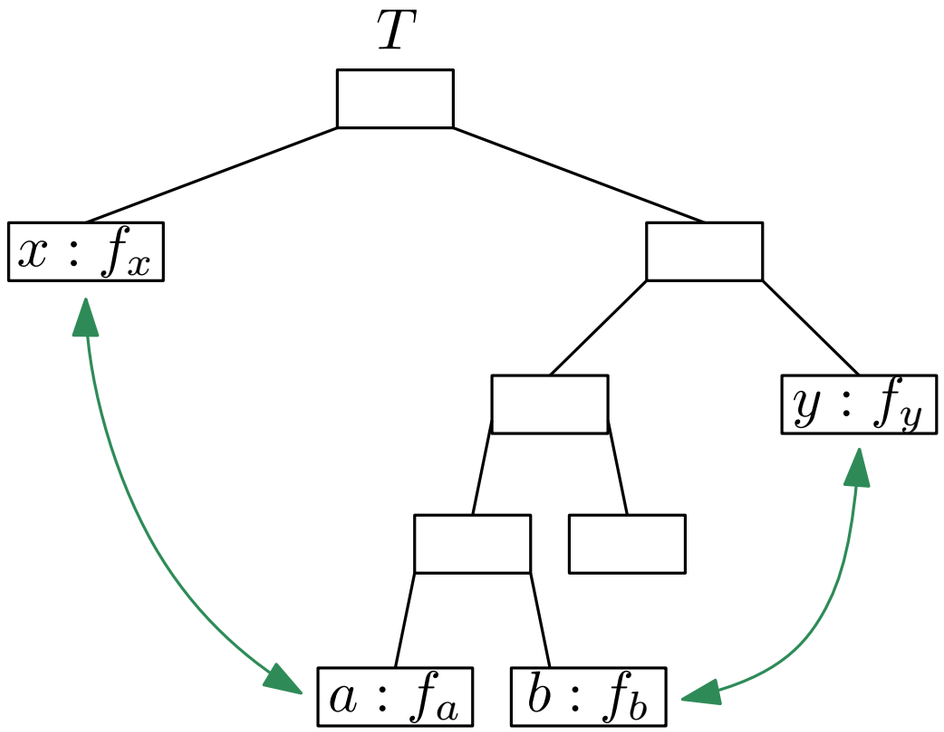
Greedy choice property

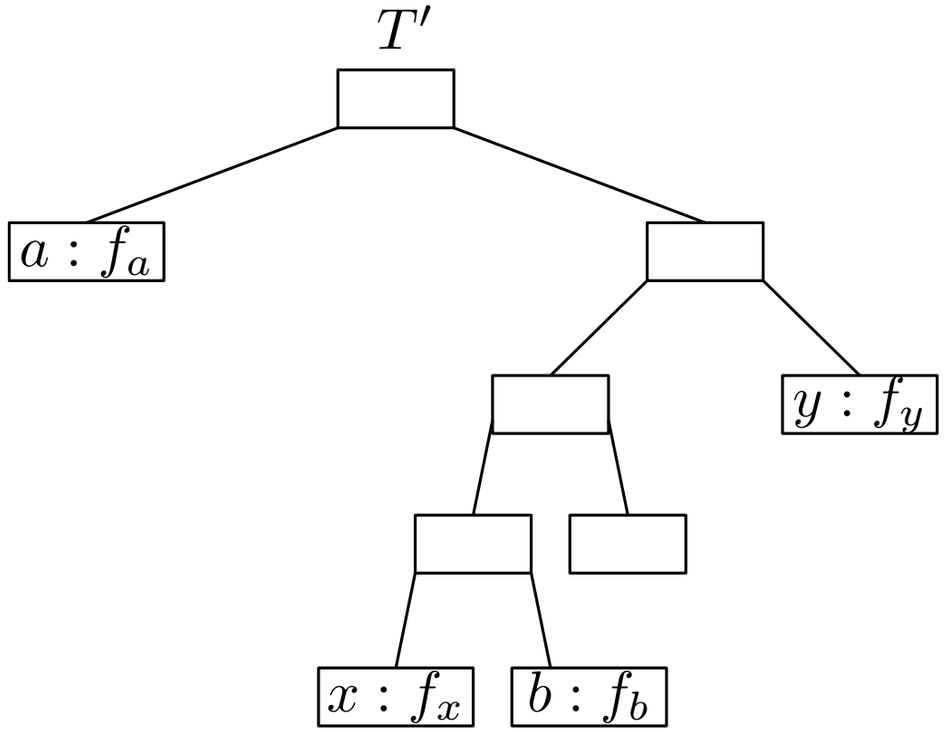
Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$

$B(T') = B(T) - f_a \cdot d_T(a) - f_x \cdot d_T(x) + f_a \cdot d_T(x) + f_x \cdot d_T(a)$





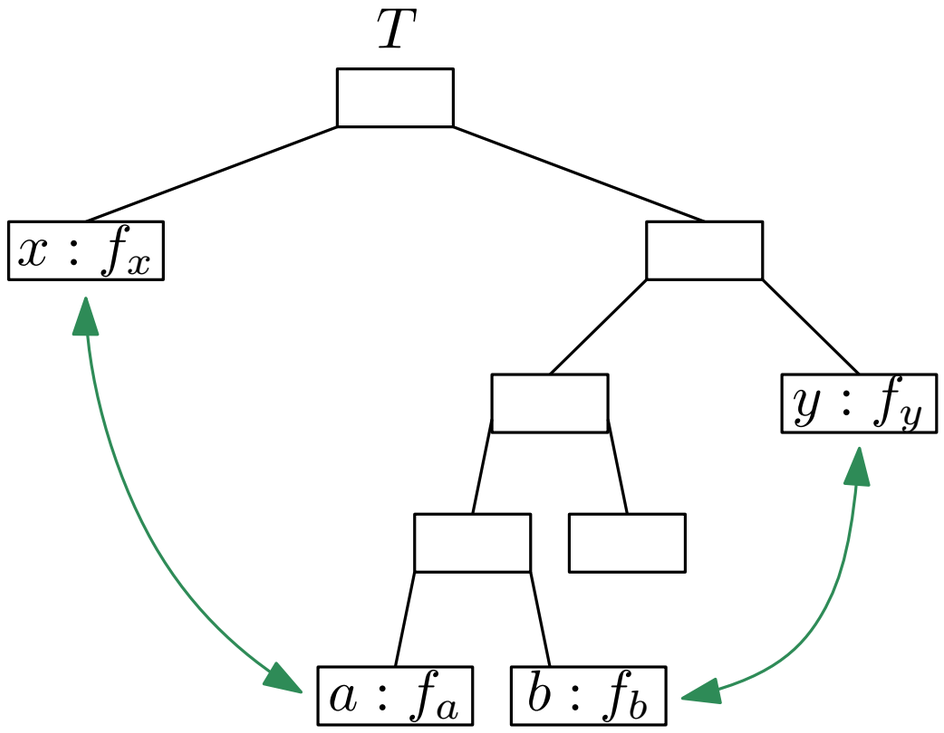
Greedy choice property

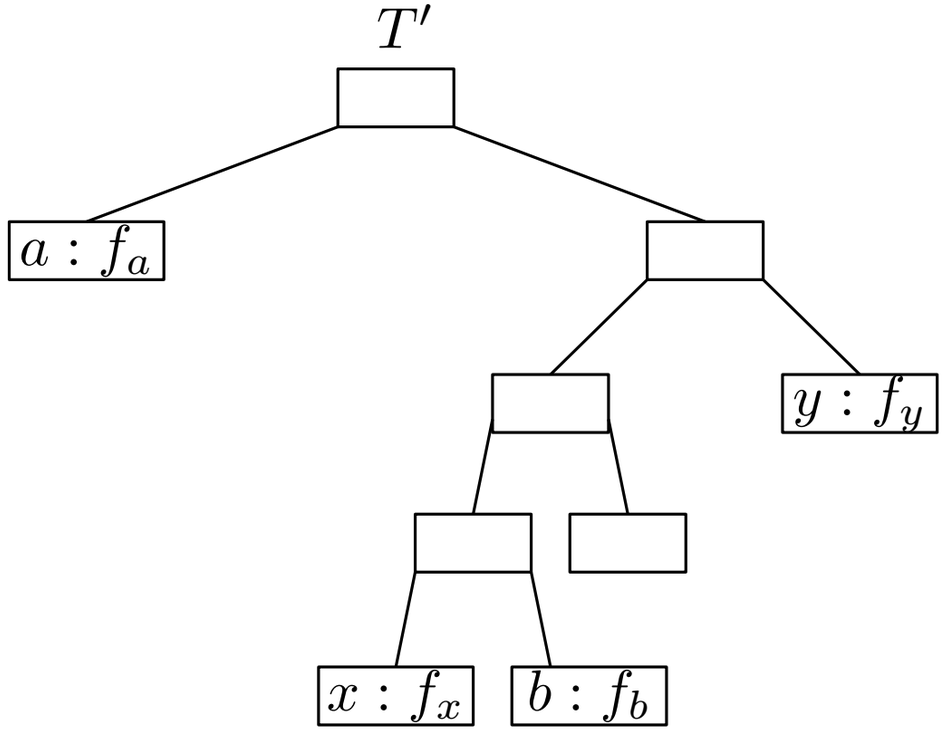
Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$

$$\begin{aligned} B(T') &= B(T) - f_a \cdot d_T(a) - f_x \cdot d_T(x) + f_a \cdot d_T(x) + f_x \cdot d_T(a) \\ &= B(T) + (f_x - f_a) \cdot (d_T(a) - d_T(x)) \end{aligned}$$





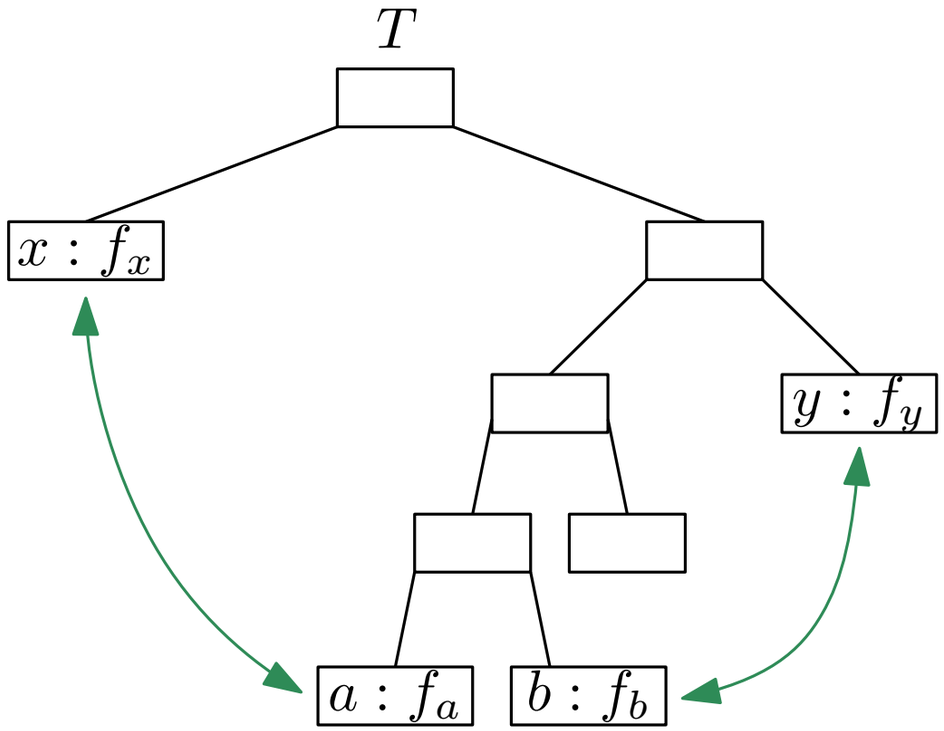
Greedy choice property

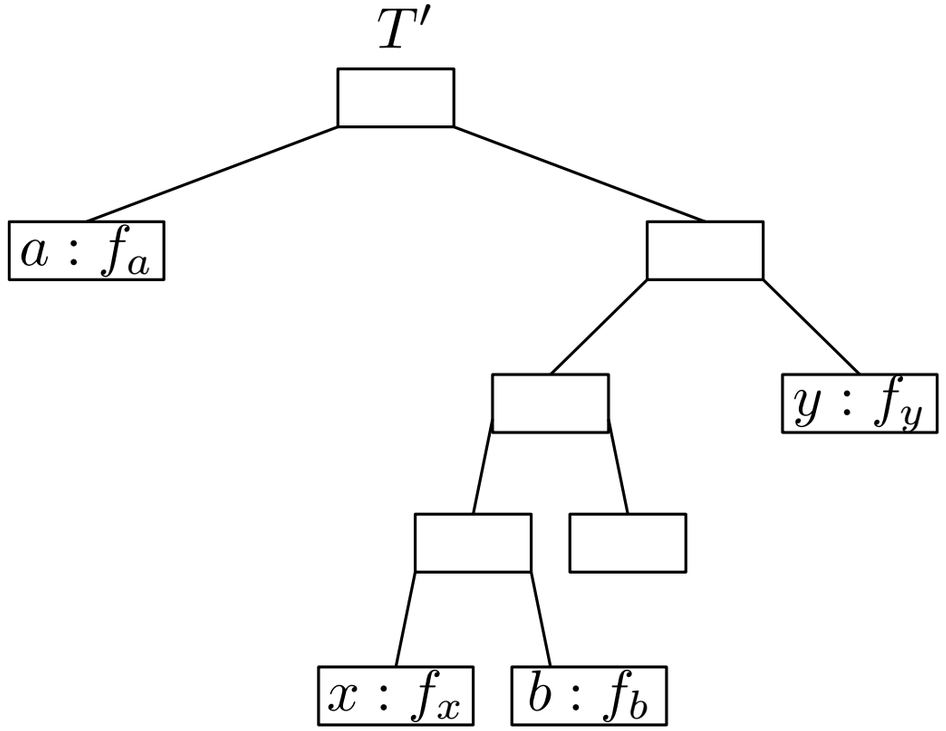
Lad x og y være bogstaver med lavest frekvens. Der findes optimalt parse tree hvor x og y er søskende (og har maksimum dybde).

$f_c = c.freq$

$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$

$$B(T') = B(T) - f_a \cdot d_T(a) - f_x \cdot d_T(x) + f_a \cdot d_T(x) + f_x \cdot d_T(a)$$
$$= B(T) + \underbrace{(f_x - f_a)}_{\leq 0} \cdot \underbrace{(d_T(a) - d_T(x))}_{\geq 0} \leq B(T)$$





Optimalitet

Induktion over n. Basis: $n = 2$, trivielt.

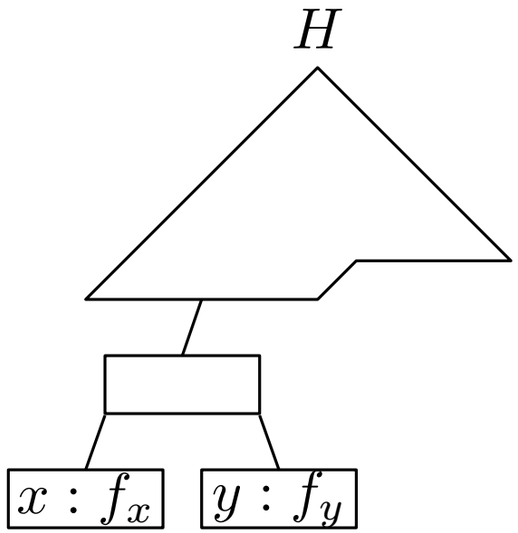
Antag at algoritmen giver optimalt parse tree når $\mid C \mid = n - 1$ og betragt alfabet med størrelse n.

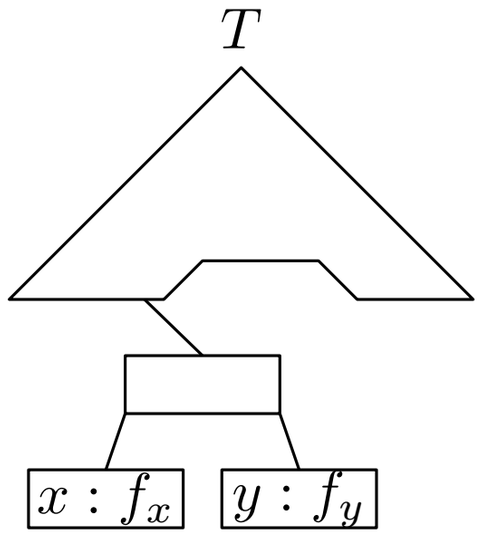
x, y: bogstaver med lavest frekvens.

T: optimalt parse tree hvor x og y er naboer.

H: resultat af algoritme.

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$





Optimalitet

Induktion over n. Basis: $n = 2$, trivielt.

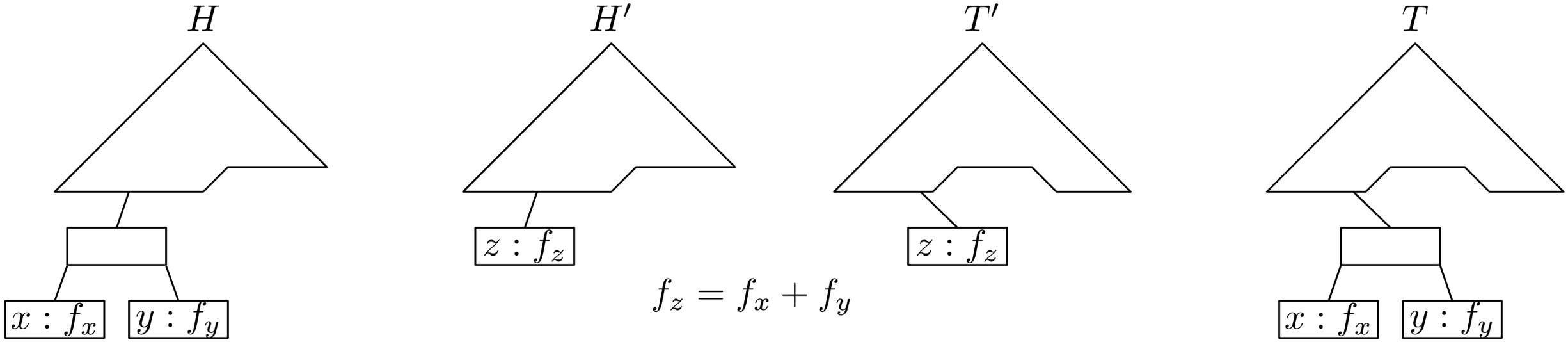
Antag at algoritmen giver optimalt parse tree når $\mid C \mid = n - 1$ og betragt alfabet med størrelse n.

x, y: bogstaver med lavest frekvens.

T: optimalt parse tree hvor x og y er naboer.

H: resultat af algoritme.

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$



$$f_z = f_x + f_y$$

Optimalitet

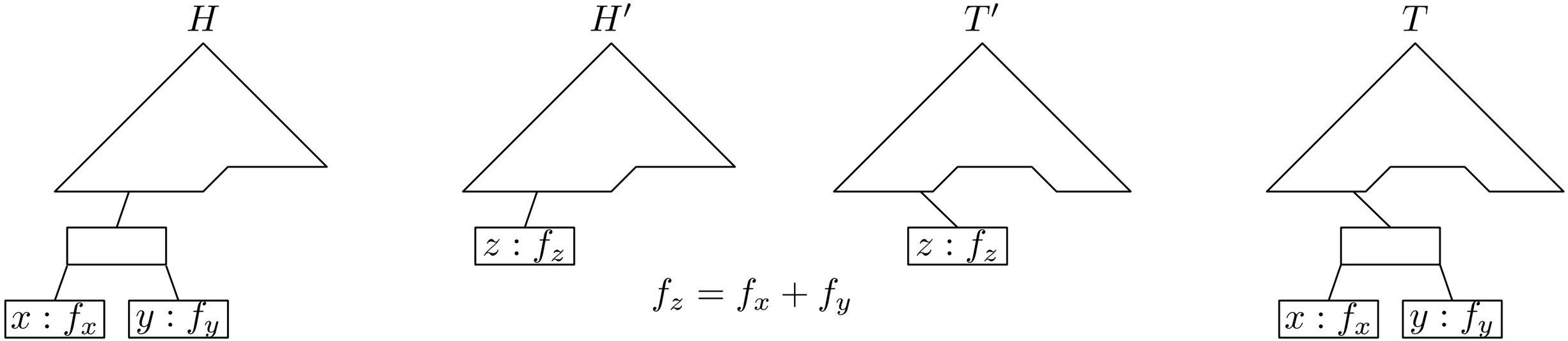
Induktion over n. Basis: $n = 2$, trivielt.

Antag at algoritmen giver optimalt parse tree når $\mid C \mid = n - 1$ og betragt alfabet med størrelse n.

x, y: bogstaver med lavest frekvens.

T: optimalt parse tree hvor x og y er naboer.

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$



$$f_z = f_x + f_y$$

$$B(H) = \ldots + (f_x + f_y) \cdot d_H(x)$$

$$B(H') = \ldots + (f_x + f_y) \cdot (d_H(x) - 1)$$

Optimalitet

Induktion over n. Basis: $n = 2$, trivielt.

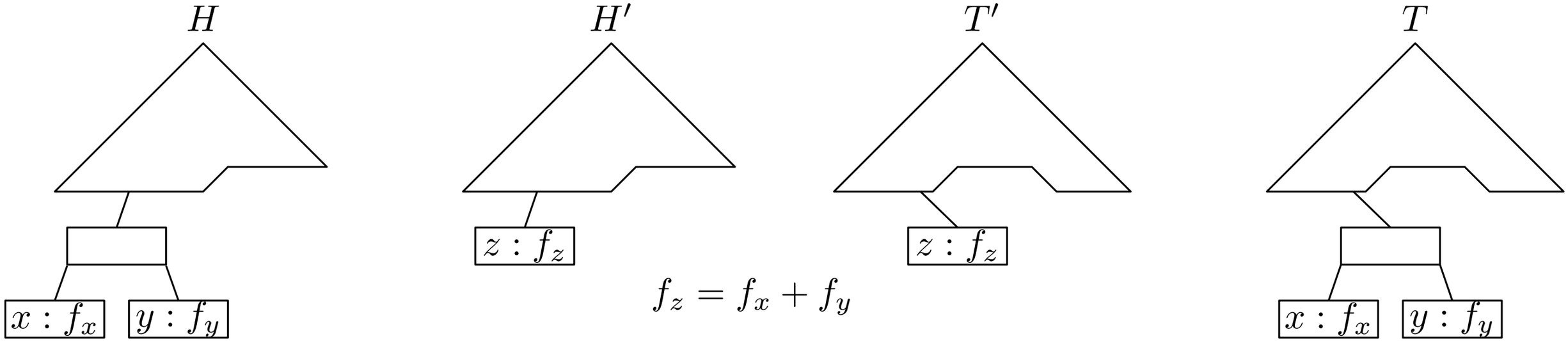
Antag at algoritmen giver optimalt parse tree når $\mid C \mid = n - 1$ og betragt alfabet med størrelse n.

x, y: bogstaver med lavest frekvens.

T: optimalt parse tree hvor x og y er naboer.

H: resultat af algoritme.

$$B(T) = \sum_{c \in C} f_c \cdot d_T(c)$$



$$f_z = f_x + f_y$$

$$B(H) = \ldots + (f_x + f_y) \cdot d_H(x)$$

$$B(H') = \ldots + (f_x + f_y) \cdot (d_H(x) - 1)$$

induktionshyp.

$$\left.\begin{array}{l} B(H) = B(H') + f_x + f_y \leq B(T') + f_x + f_y = B(T) \\ B(T) \leq B(H) \end{array}\right] \Rightarrow B(T) = B(H)$$