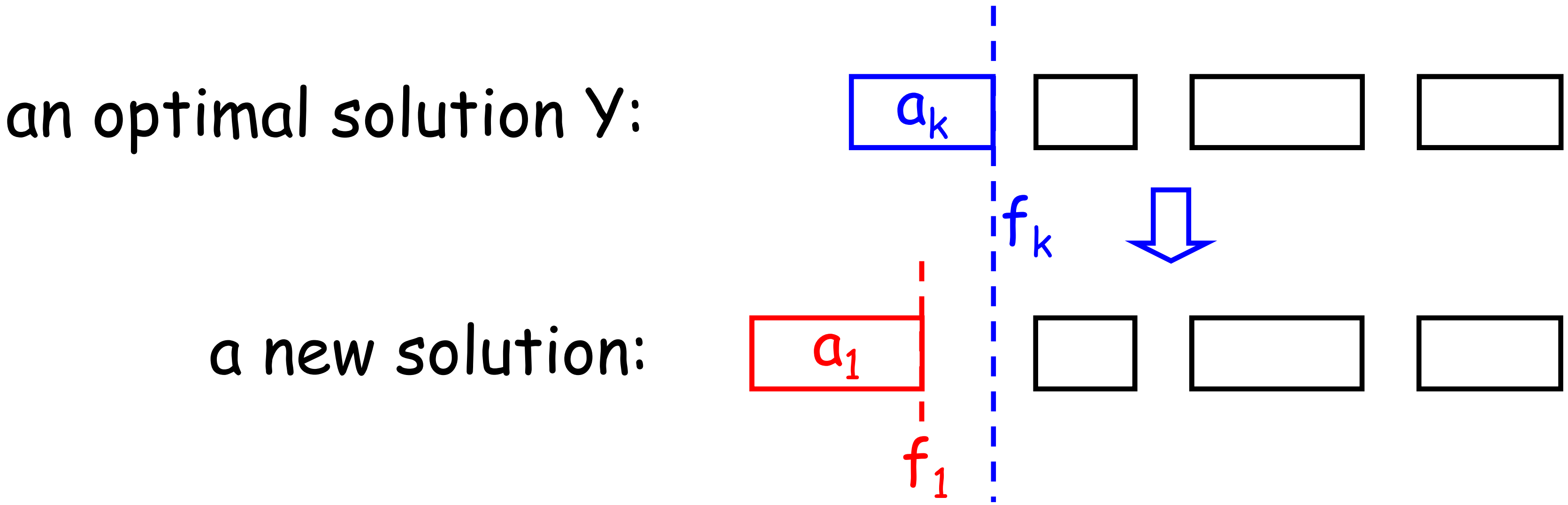
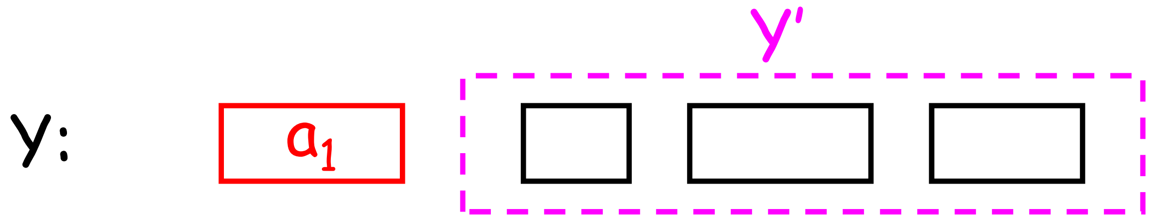
(1) Taking a_1 is correct (greedy-choice property)

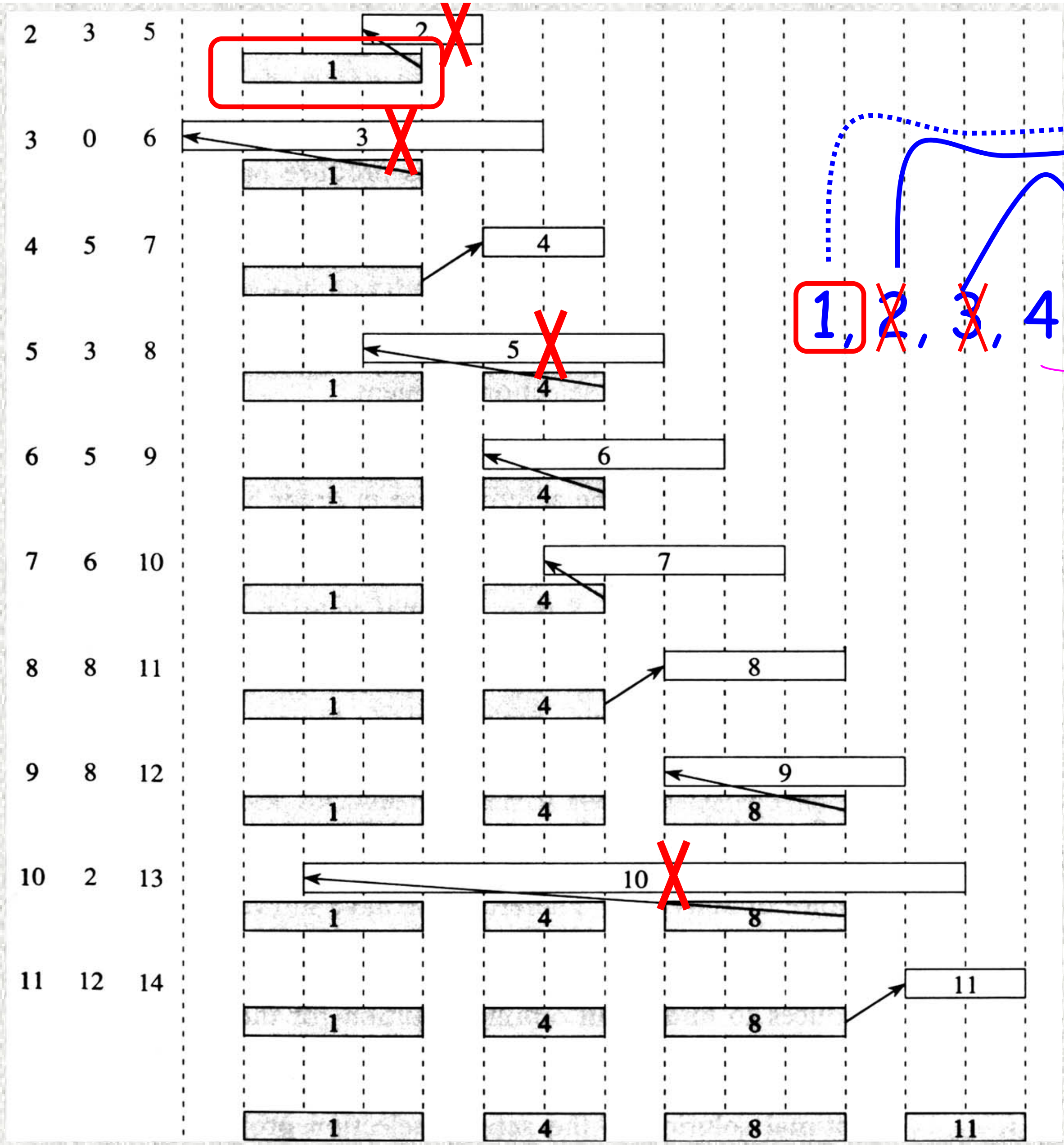


(2) Optimal substructure



Let $X = \{a_i \mid s_i < f_1\}$ and $A' = A - X = \{a_i \mid s_i \geq f_1\}$.

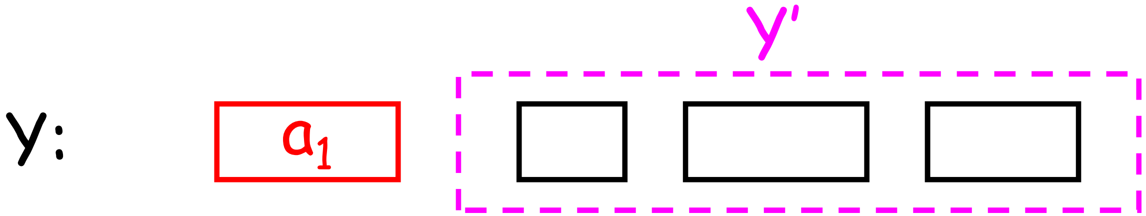
和 a_1 衝 突 　　　　和 a_1 沒 衝 突



X

$\boxed{1}, \cancel{2}, \cancel{3}, \underbrace{4, \cancel{5}, 6, 7, 8, 9, \cancel{10}, 11}_{A'}$

(2) Optimal substructure



Let $X = \{a_i \mid s_i < f_1\}$ and $A' = A - X = \{a_i \mid s_i \geq f_1\}$.

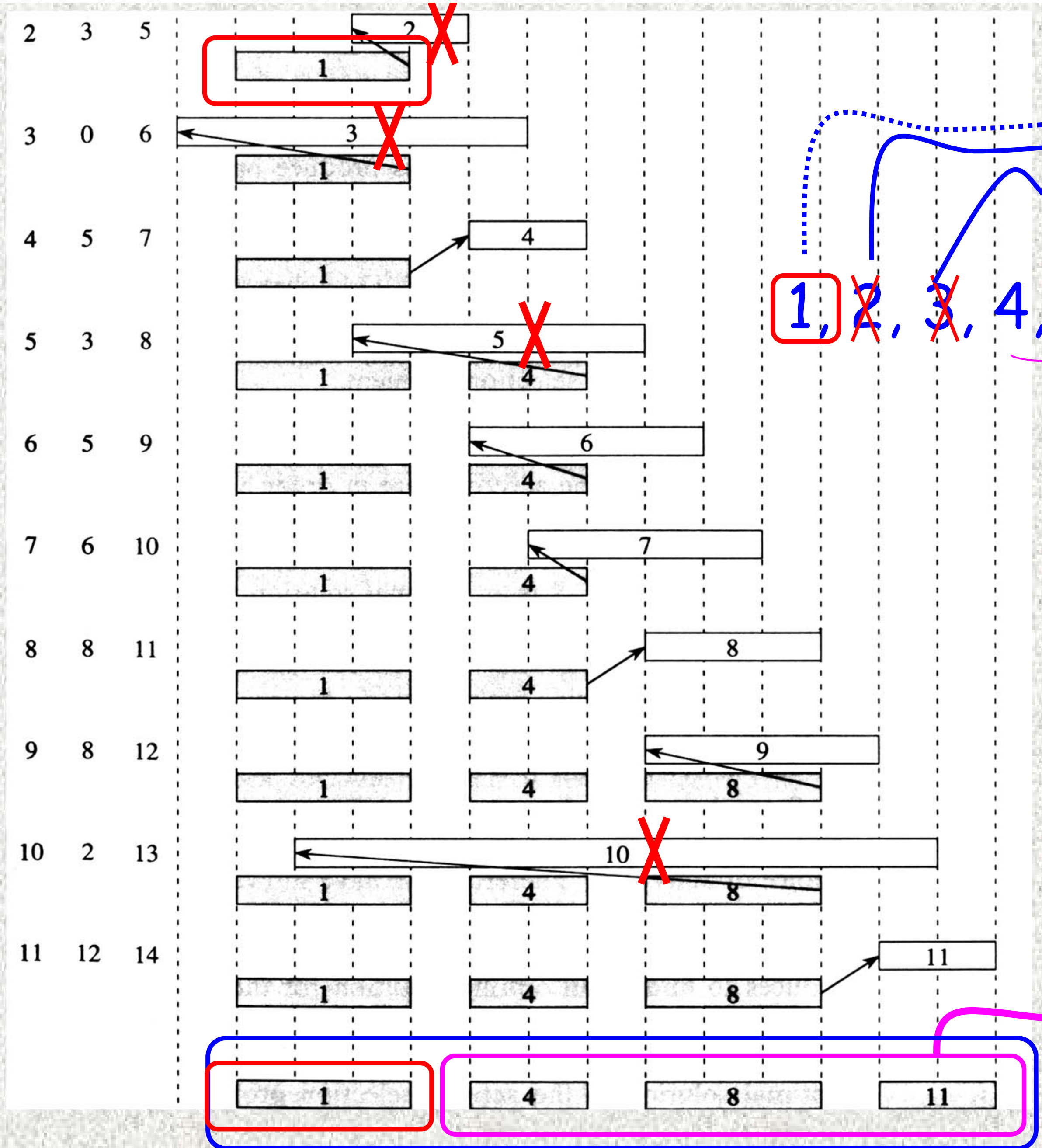
和 a_1 衝突　　和 a_1 沒衝突

After taking a_1

(i) all a_i in X should be discarded;

(ii) the problem becomes to select a maximum set of compatible activities in A'

$\Rightarrow$ Y' is optimal for A'



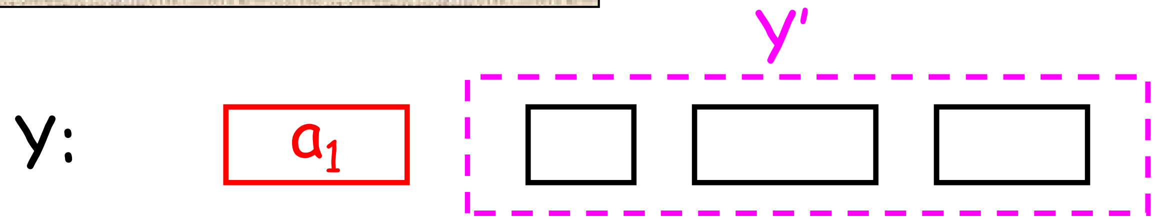
X

$\boxed{1}, \cancel{2}, \cancel{3}, \underbrace{4, \cancel{5}, 6, 7, 8, 9, \cancel{10}, 11}_{A'}$

Y': optimal for A'

Y: optimal for A

(2) Optimal substructure



Let $X = \{a_i \mid s_i < f_1\}$ and $A' = A - X = \{a_i \mid s_i \geq f_1\}$.

和 a_1 衝突　　和 a_1 沒衝突

After taking a_1

(i) all a_i in X should be discarded;

(ii) the problem becomes to select a maximum set of compatible activities in A'

$\Rightarrow$ Y' is optimal for A'

(after a choice $\rightarrow$ same problem of smaller size)

Optimal substructure

$A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$

$W = \{7, 12, 9, 6, 11, 15, 12, 7\}$

0-1:

optimal for $\begin{cases} A \\ C = 40 \end{cases}$

a_1, w_1

a_3, w_3

a_4, w_4

a_6, w_6

optimal for $\begin{cases} A' = \{a_1, a_2, a_3, a_4, a_5, \cancel{a_6}, \cancel{a_7}, \cancel{a_8}\} \\ C' = 40 - 15 = 25 \end{cases}$

$w_6 = 15$

fractional:

optimal for $\begin{cases} A \\ C = 40 \end{cases}$

a_2, x_2

a_3, x_3

a_5, x_5

a_7, x_7

optimal for $\begin{cases} A' = \{a_1, a_2, a_3, a_4, a_5, a_6, \cancel{a_7}, \cancel{a_8}\} \\ C' = 40 - 10 = 30 \end{cases}$

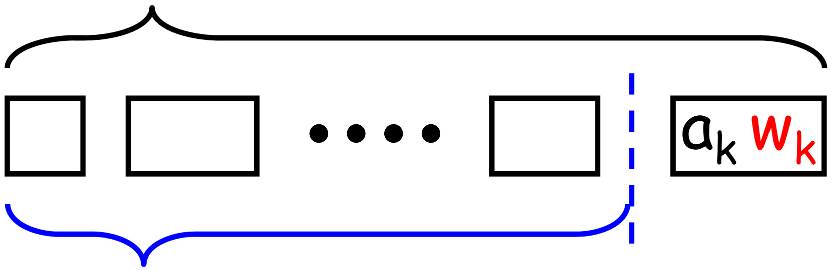
$x_7 = 10$

$(x_i \le w_i)$

Optimal substructure

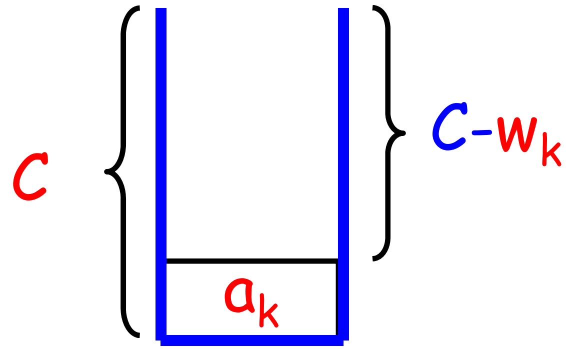
0-1:

optimal for $\begin{cases} A = \{a_1, a_2, \ldots, a_n\} \\ C \end{cases}$



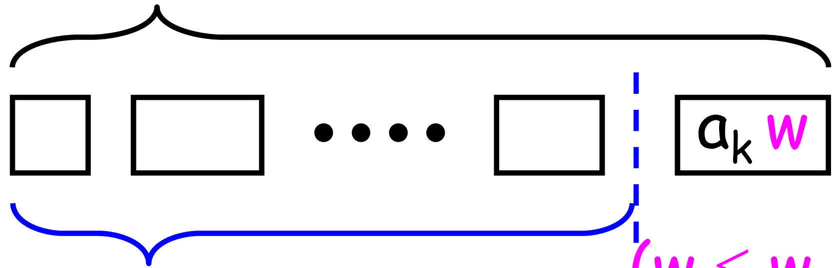
optimal for

$$\begin{cases} A' = \{a_1, a_2, \ldots, a_{k-1}, \cancel{a_k}, \ldots, \cancel{a_n}\} \\ C' = C - w_k \end{cases}$$



fractional:

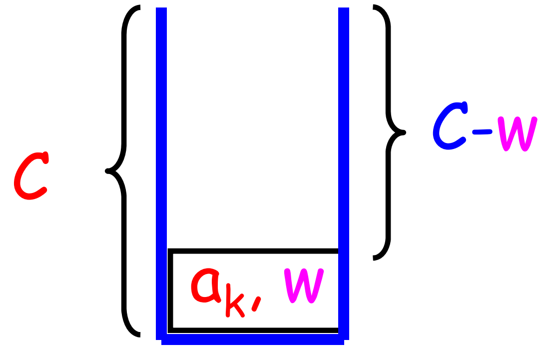
optimal for $\begin{cases} A = \{a_1, a_2, \ldots, a_n\} \\ C \end{cases}$



$(w \leq w_k)$

optimal for

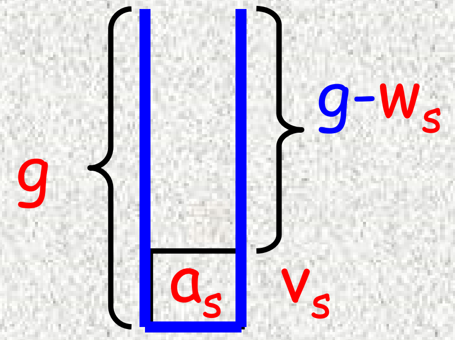
$$\begin{cases} A' = \{a_1, a_2, \ldots, a_{k-1}, \cancel{a_k}, \ldots, \cancel{a_n}\} \\ C' = C - w \end{cases}$$



A naive DP: 0/1 knapsack

* $f(g, k)$: optimal value for $\begin{cases} a_1\, a_2 \ldots a_{s-1}\, a_s \ldots a_k \\ \text{capacity is } g \end{cases}$

* solution: $f(C,n)$



$$f[g, k] = \max_{\substack{1 \le s \le k \\ w_s \le g}} \left\{ f[g-w_s, s-1] + v_s \right\}$$

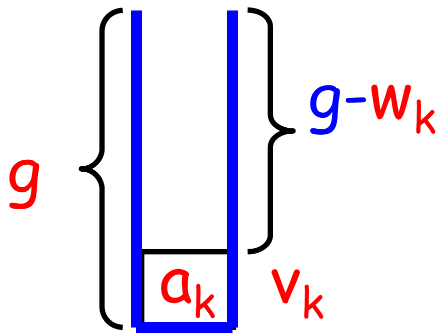
Time: $O(Cn^2)$

0-1 Knapsack problem

(integer weights, DP)

* $f(g,k)$: optimal value for
$\begin{cases} a_1\, a_2 \ldots a_k \\ \text{capacity is } g \end{cases}$

* solution: $f(C,n)$



* $f(g,k) = \max \begin{Bmatrix} f(g,\ k-1) \\ f(g-w_k,\ k-1) + v_k \end{Bmatrix}$

* $f(0,k) = f(g,0) = 0$, $f(-,k) = -\infty$

* Time: $O(Cn)$

* optimal substructure

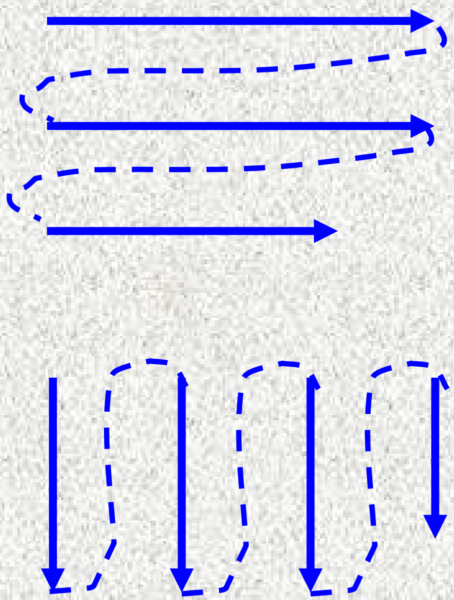
Case 1. a_k is not selected

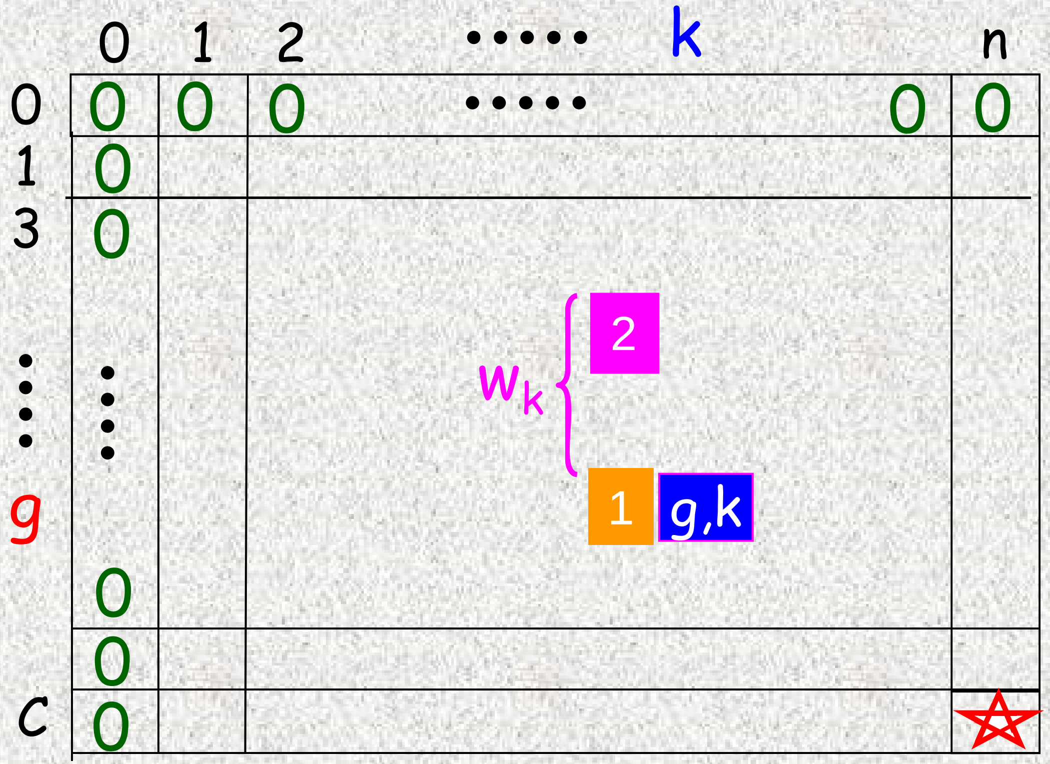
$$\underbrace{\boxed{a_1}\ \boxed{a_2}\ \cdots\cdots\ \boxed{a_{k-1}}}_{f(g,k-1)}\ \boxed{\cancel{a_k}}$$

Case 2. a_k is selected

$$\underbrace{\boxed{a_1}\ \boxed{a_2}\ \cdots\cdots\ \boxed{a_{k-1}}}_{f(g-w_k,k-1)}\ \underbrace{\boxed{a_k}}_{v_k}$$

* $f(g,k) = \max\{\ \underbrace{f(g,\ k-1)}_{1},\ \underbrace{f(g-w_k,\ k-1)}_{2} + v_k\ \}$
* goal: $f(C,n)$

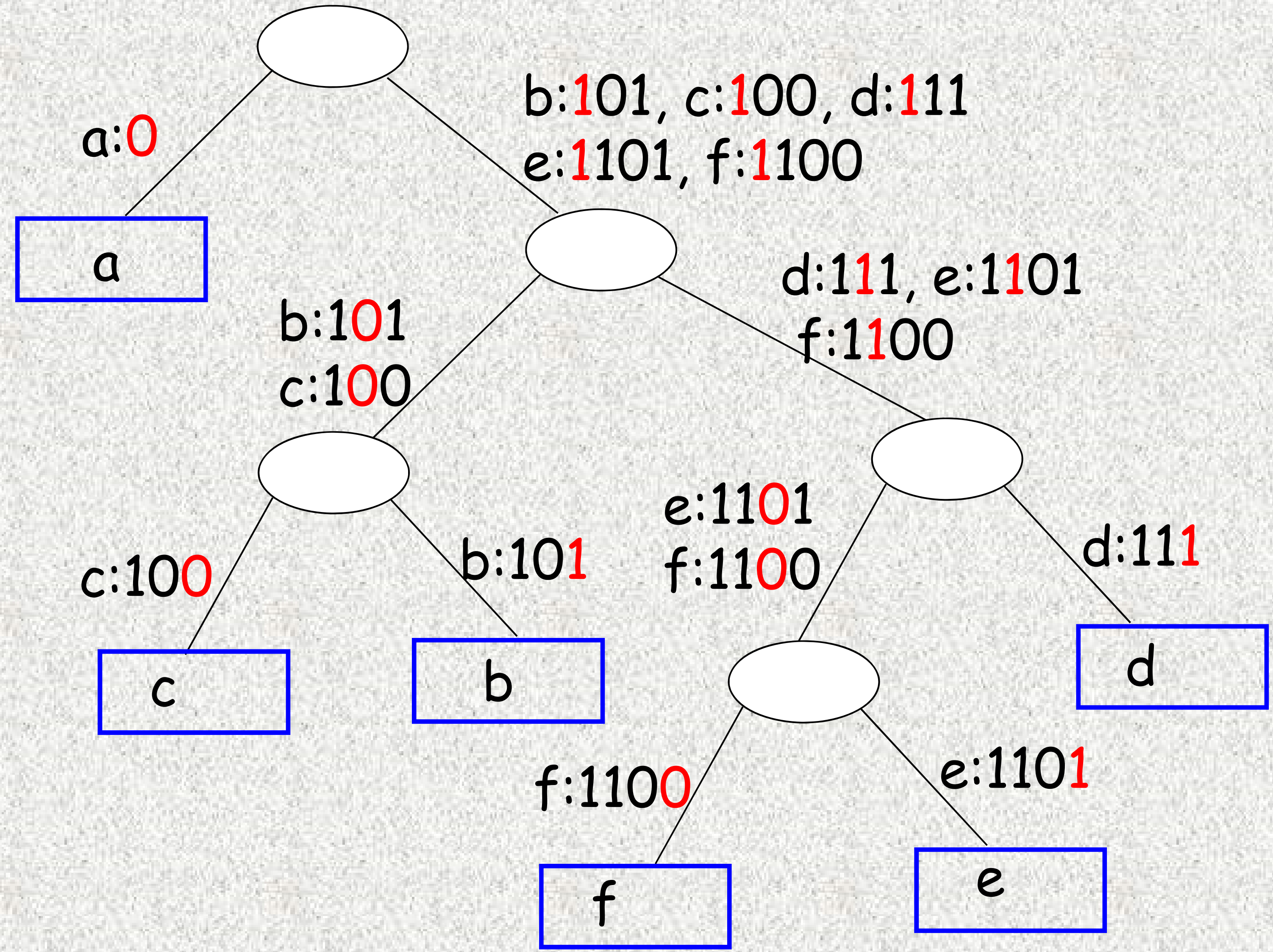




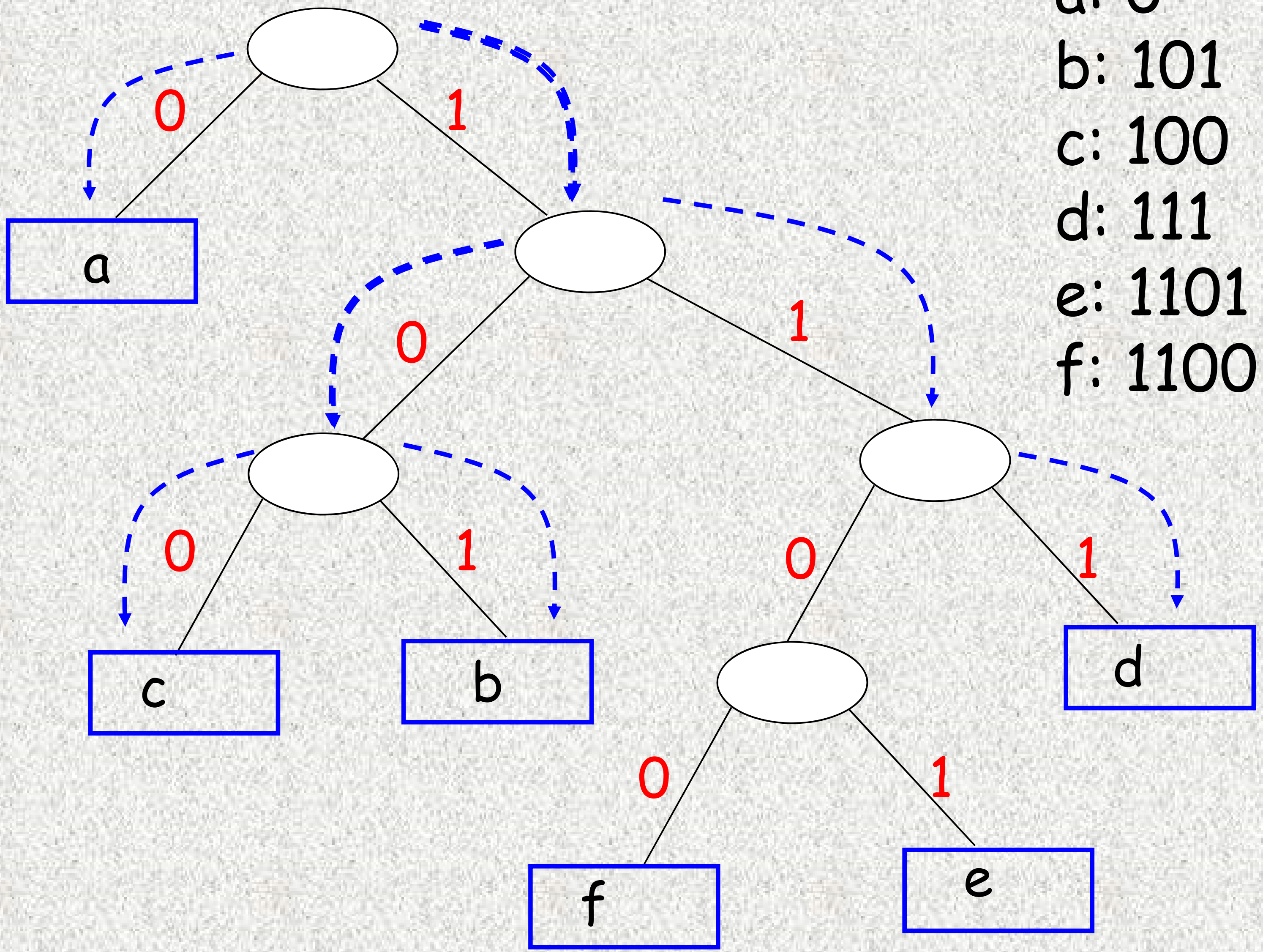
Time: $\underbrace{Cn}_{\text{table size}} \times O(1) = O(Cn)$

prefix code ⇨ n-leaf full binary tree

a:0, b:101, c:100, d:111, e:1101, f:1100



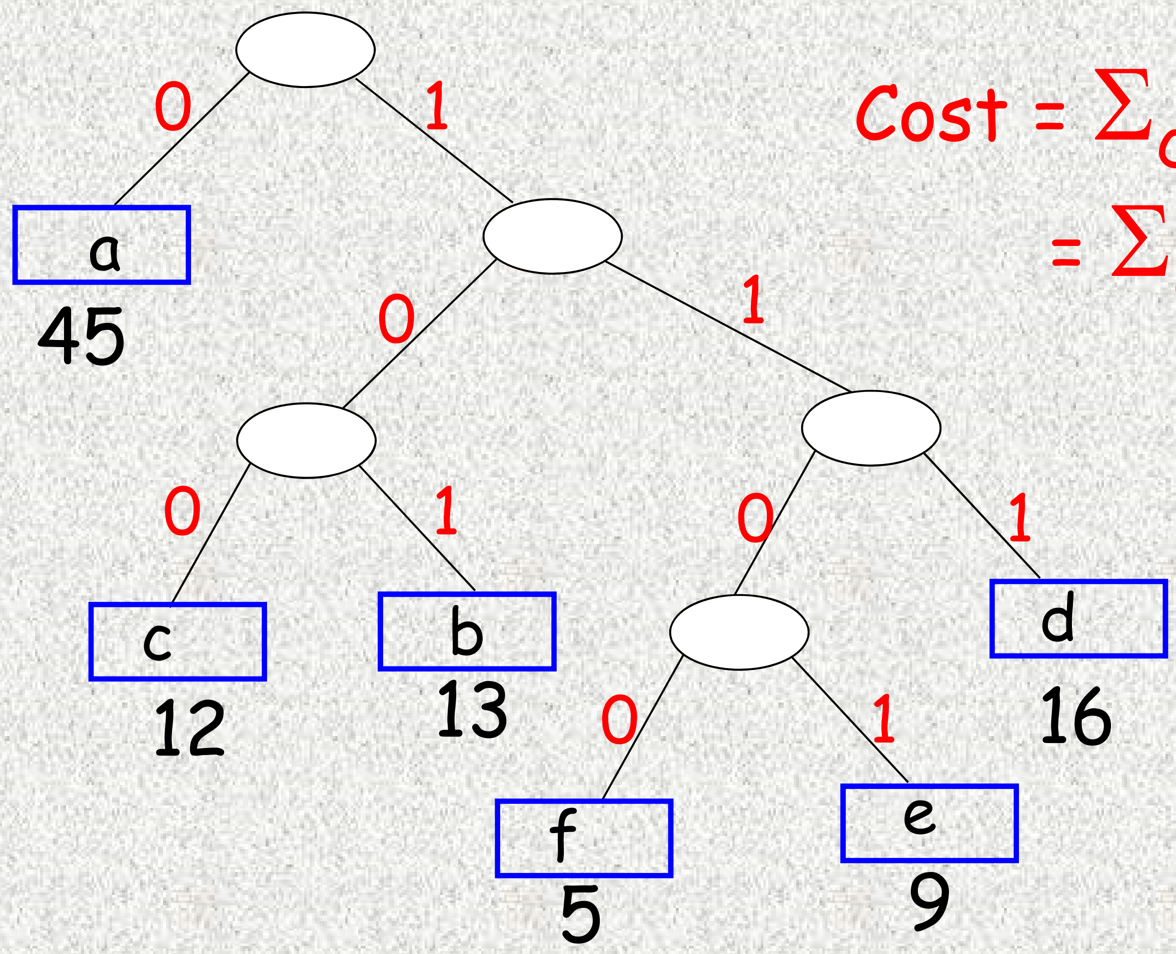
n-leaf tree ⇨ prefix code



a: 0
b: 101
c: 100
d: 111
e: 1101
f: 1100

a: 0   b: 101   c: 100   d: 111   e: 1101   f: 1100

(45)   (13)   (12)   (16)   (9)   (5)

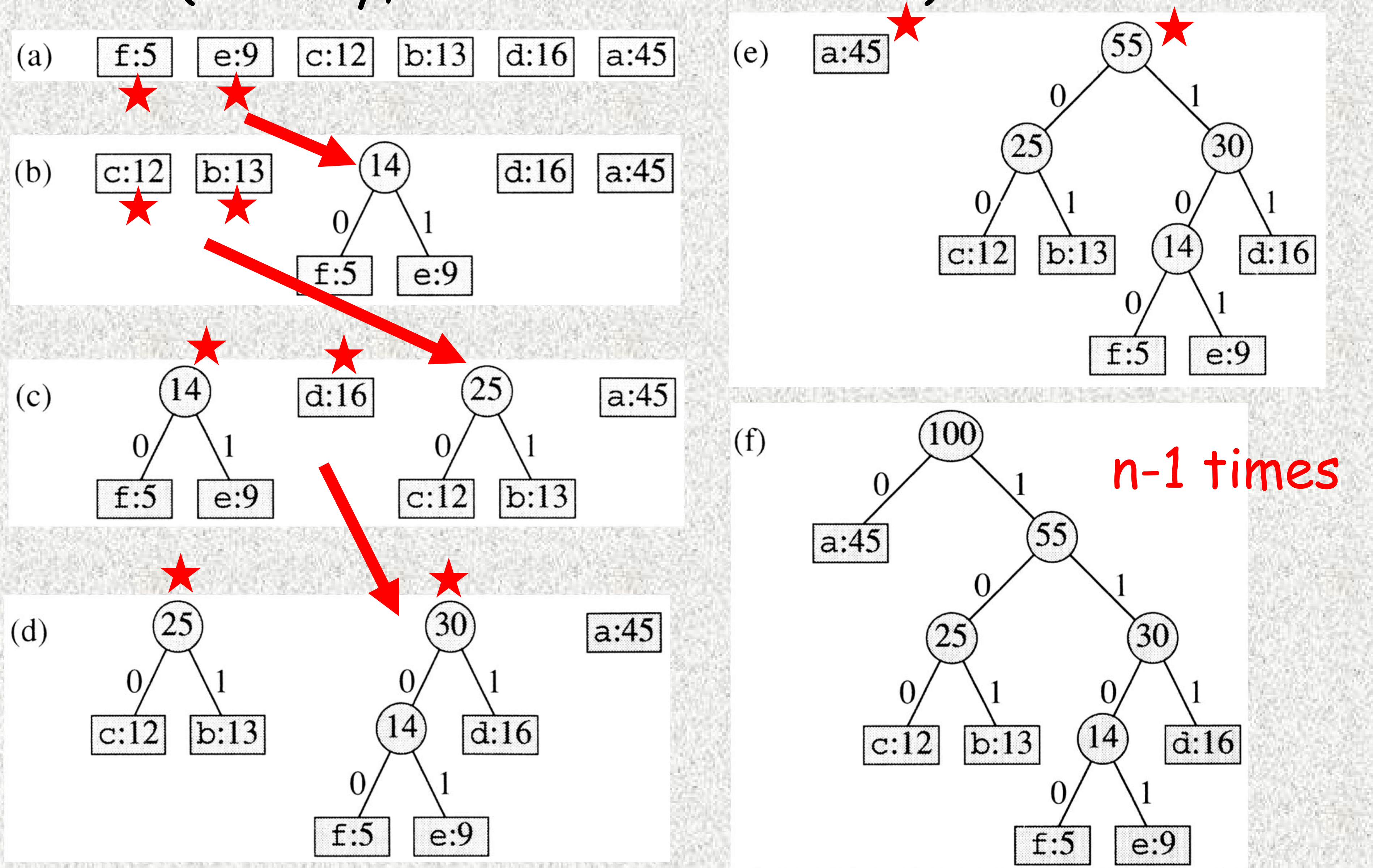


$$\text{Cost} = \sum_c f(c) \times \mathbf{len(c)}$$

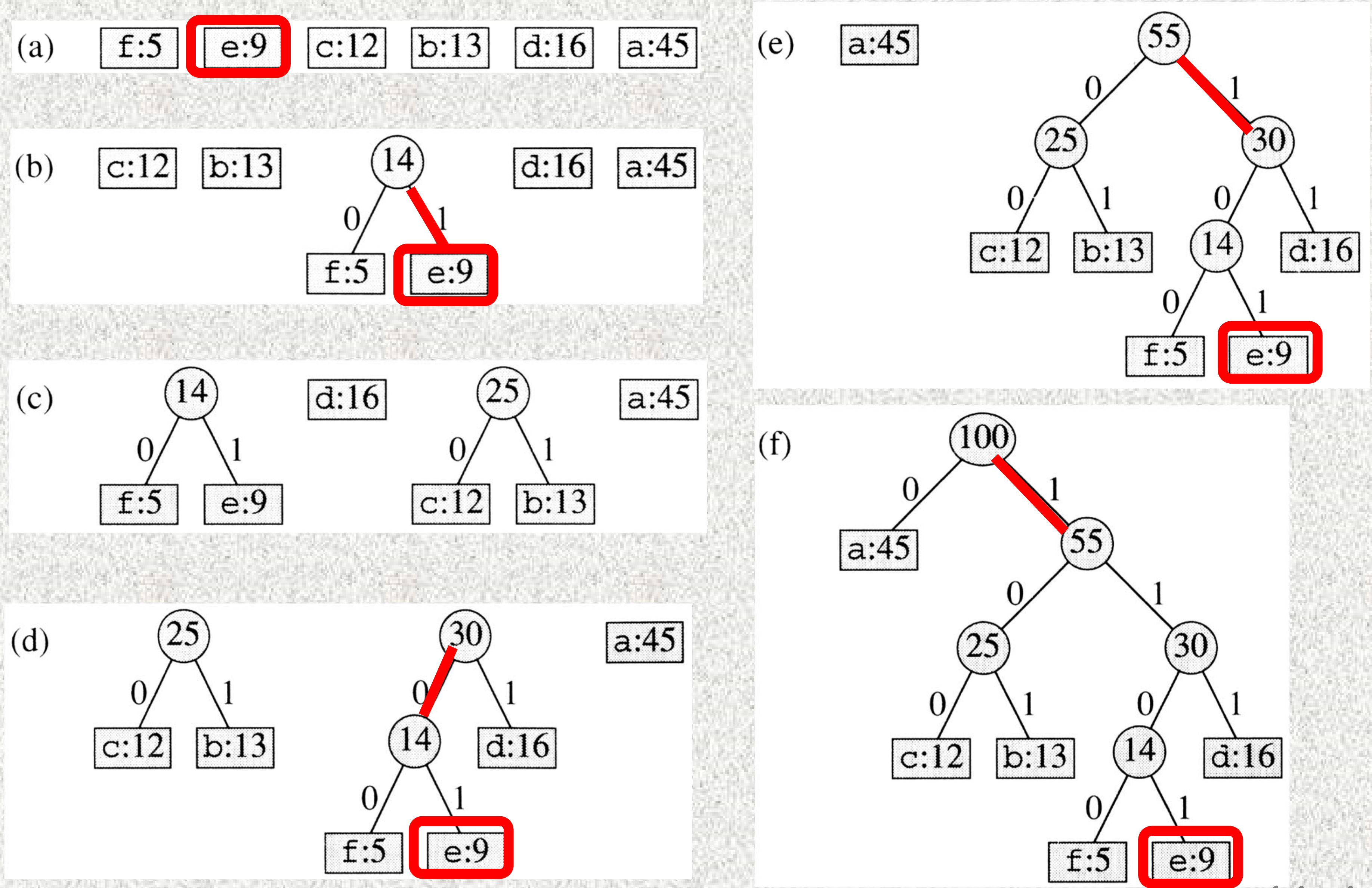
$$= \sum_c f(c) \times \mathbf{depth(T,c)}$$

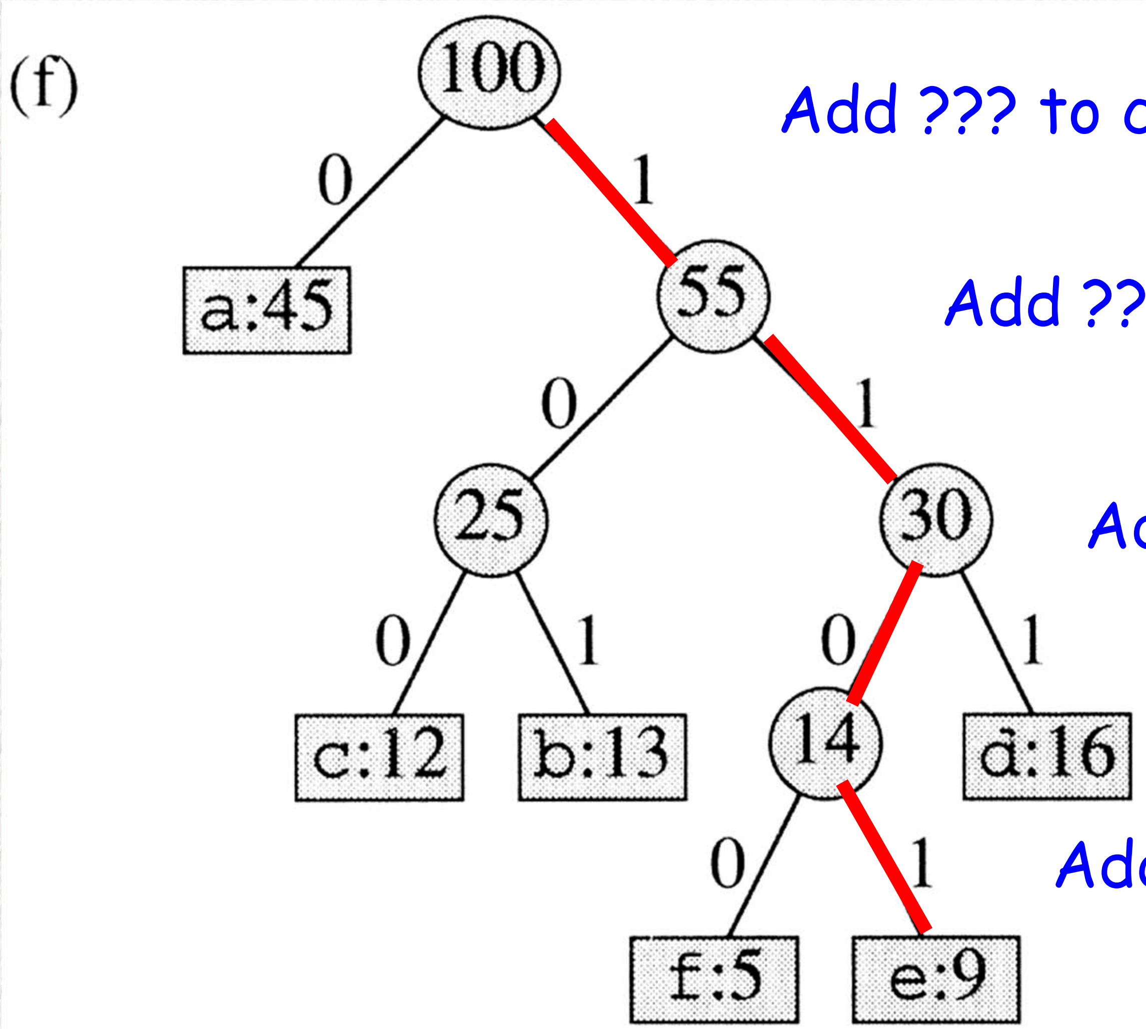
leaves

Algo Outline: repeatedly combine two trees into one
(initially, each leaf is a "trees)



e : 4 "combining", e contributes 4 x 9 to cost
(each combining +9 to cost)



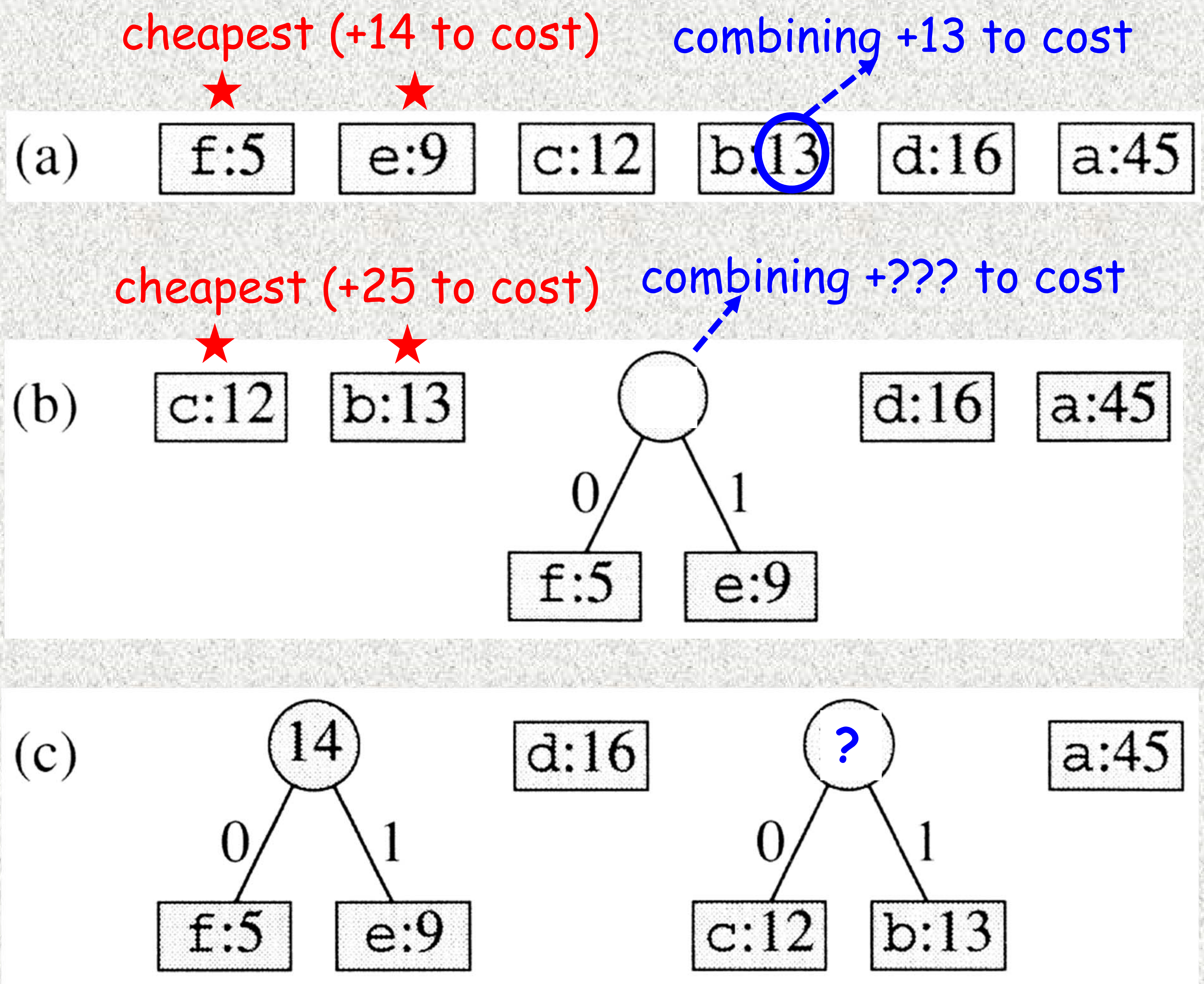


Add ??? to cost

Add ??? to cost

Add ??? to cost

Add ??? to cost


cheapest (+14 to cost)
combining +13 to cost
(a)
f:5
e:9
c:12
b:13
d:16
a:45
cheapest (+25 to cost)
combining +??? to cost
(b)
c:12
b:13
0
1
f:5
e:9
d:16
a:45
(c)
14
0
1
f:5
e:9
d:16
?
0
1
c:12
b:13
a:45

(See 6-8)

Priority Queue	binary heap (max-heap)
Build	O(n)
Insert	O(lg n)
Maximum	O(1)
Increase-Key	O(lg n)
Extract-Max	O(lg n)

Priority Queue	binary heap (min-heap)
Build	O(n)
Insert	O(lg n)
Minimum	O(1)
Decrease-Key	O(lg n)
Extract-Min	O(lg n)

Hint for correctness

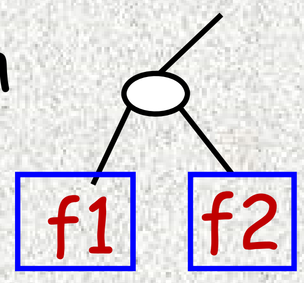
Building an optimal tree for (f1, f2, f3, f4, ..., fn)

(size = n)

(Assume sorted)

Greedy-choice property:

there is an optimal solution with



Optimal substructure:

after merge f1 and f2, the problem becomes "building a tree for (f1+f2, f3, f4, ..., fn)

(size = n - 1)