------------------------------ 16 ------------------------------

Greedy Algorithms

Greedy algorithm**:** one always makes a locally optimal choice. (in the hope that this choice will lead to a globally optimal solution)

16.1 An activity-selection problem

Input: n activities with start time s_i and finish time f_i
Output: a maximum set of compatible activities

Step 0: Sort the input according to f_i increasingly.
Step 1: Schedule the activities one by one.

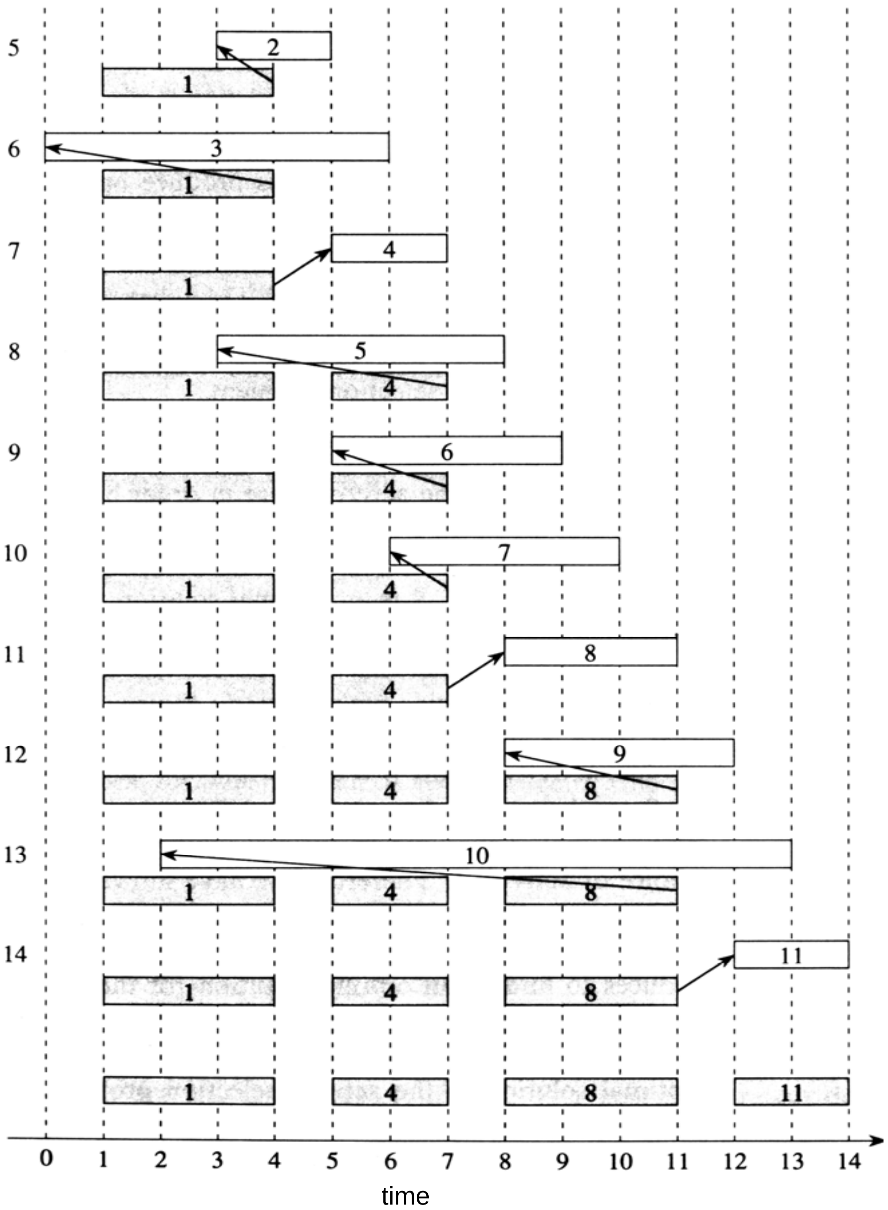
Time: $T(n) = \begin{cases} O(n) & \text{if the input is sorted,} \\ O(n\lg n) & \text{otherwise.} \end{cases}$

Correctness:
(Assume that $A=\{a_1, a_2, ..., a_n\}$ is sorted)

(1) There is an optimal solution contains a_1.
(If Y is an optimal solution, Y–{first in Y}$\cup\{a_1\}$ is also an optimal solution.)
(2) Let $Y=\{a_1\}\cup Y'$ be an optimal solution to A.
Then, Y' is an optimal solution to $A'=\{a_i | s_i \geq f_1\}$.

Example:

i	s_i	f_i
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14



16.2 Elements of Greedy strategy

1. ***Greedy-choice property*****:** a globally optimal solution can be arrived by making a locally optimal (greedy) choice. (top-down, usually)

2. ***Optimal substructure*****:** an optimal solution to the problem contains optimal solutions to sub-problems.

0-1 knapsack problem (integer)**:**

Input: *n* items with weight w_i and value v_i
capacity C (w_i, v_i, and C are integers)

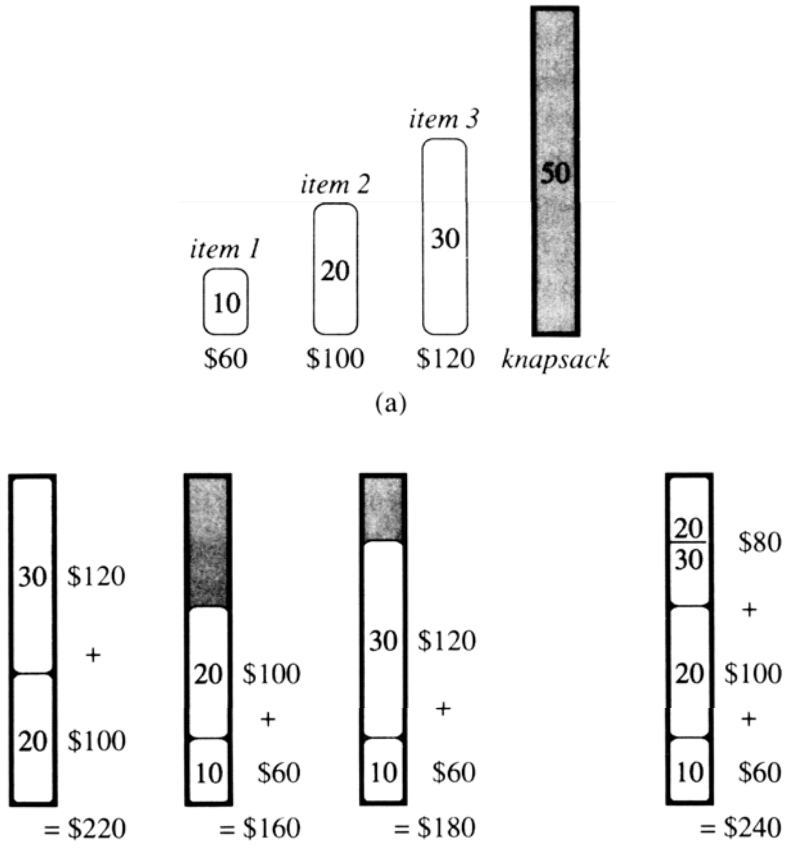
Output: a subset of items with weight$\leq C$ and maximum value
(each item must be taken or left behind)

Fractional knapsack problem**:** Same as above. But fractions of items can be taken.

- The fractional problem has both properties. (Greedy according to v_i/w_i, $O(n \lg n)$ time)

- The 0/1 problem only has the optimal-substructure property. (dynamic programming)

Example:



(b) 0-1 (c) fractional

16.3 Huffman codes (for compression)

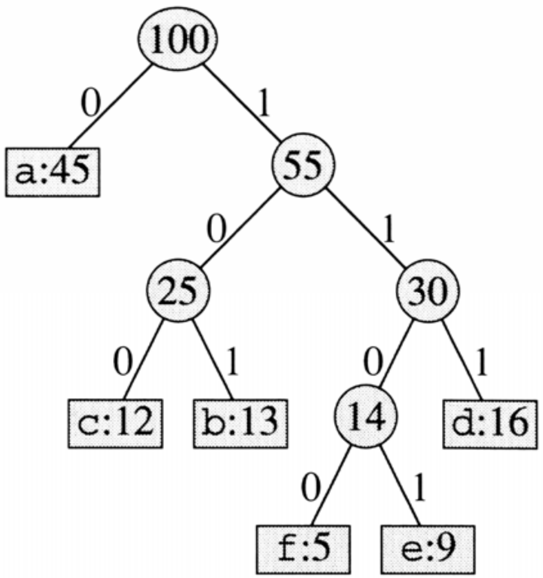
	a	b	c	d	e	f
frequency	45	13	12	16	9	5
fixed-length	000	001	010	011	100	101
variable-length	0	101	100	111	1101	1100

Cost: (fixed) $3\times(45+13+12+16+9+5)$
(var) $1\times45+3\times(13+12+16)+4\times(9+5)$ (√)

Coding: abc $\Rightarrow$ 0·101·100 $\Rightarrow$ 0101100
Decoding: 001011101 $\Rightarrow$ 0·0·101·1101 $\Rightarrow$ aabe

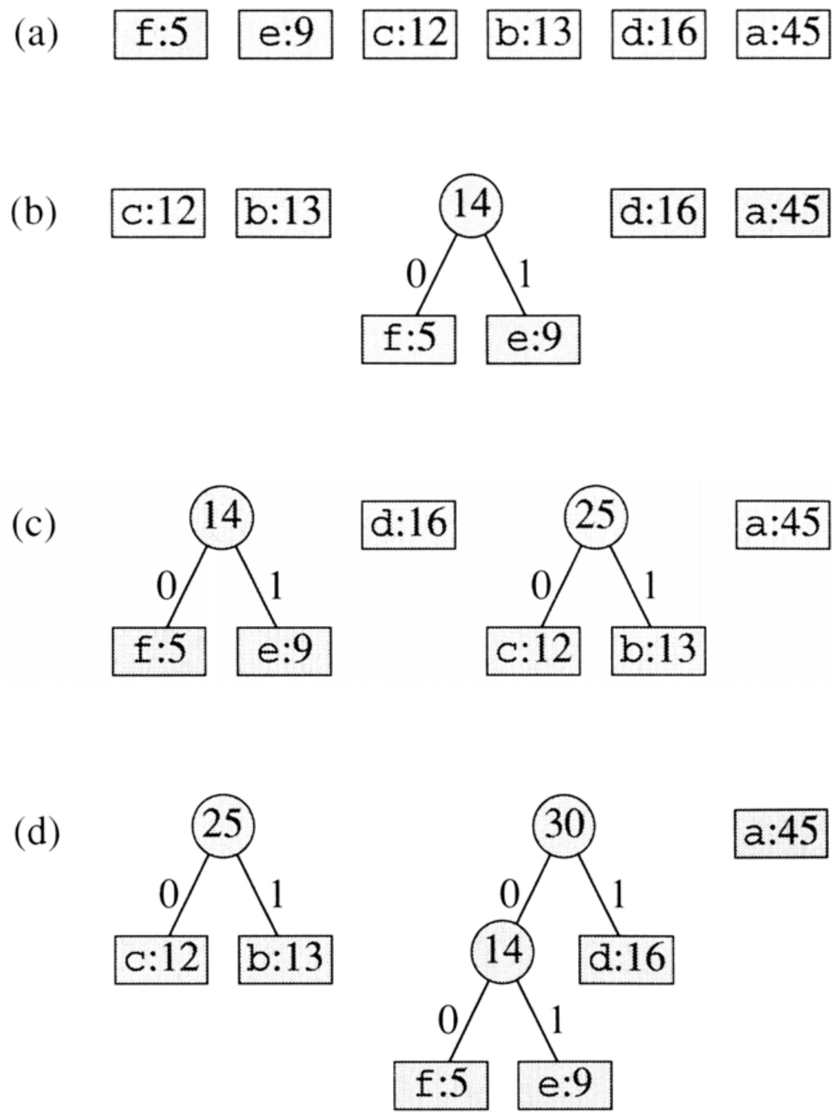
***Prefix codes*:** no codeword is a prefix of other
(<a:00, b:0, c:10, d:1> are not prefix codes)

Representing prefix codes by a tree:

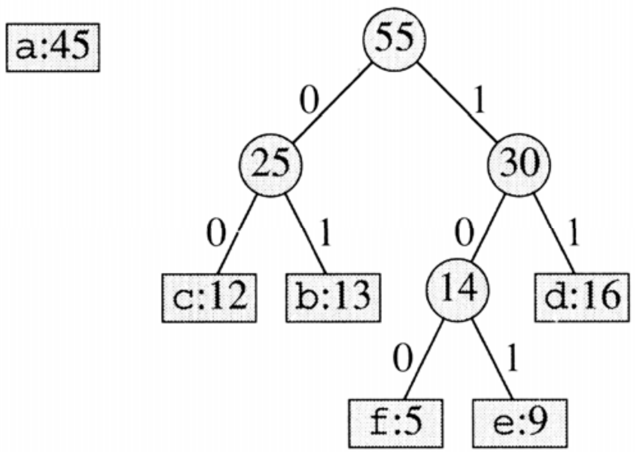


- a:0, b:101, c:100, d:111, e:1101, f:1100
- cost of T: $\sum_{c\in C} f(c)\times depth(T,c)$

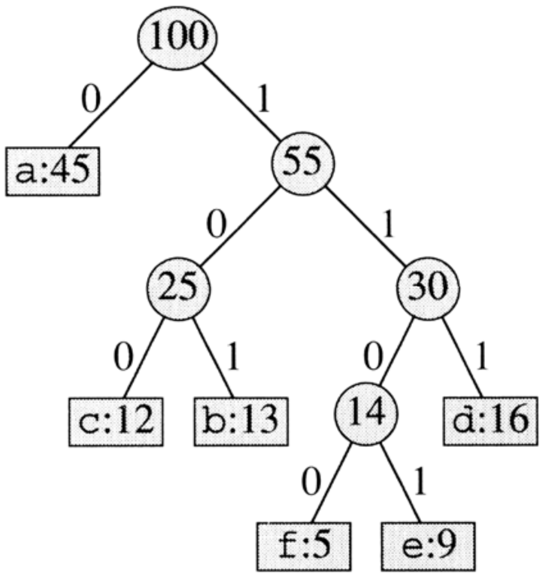
Huffman code



(e)



(f)



- Time: $O(n \lg n)$
  (using a heap as a priority queue)
- Correctness: omitted

Homework: Ex. 16.1-4, 16.2-2, 16.2-3, 16.2-4, 16.2-5, 16.2-7, 16.3-6