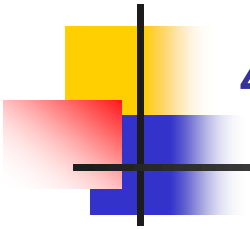


4.4 Huffman Code

- Data compression
  - **find an efficient method for encoding a data file.**
- Example coding
  - **Our character set is {a, b, c}**
  - **Our file is ababcbbbc**
  - **Fixed-length binary code: a: 00 b: 01 c: 11.**
    - **Our encoding is 000100011101010111(18 bits)**
  - **Variable-length binary code: a: 10 b: 0 c: 11 (code4.2)**
    - **Our encoding is 100100100011(13bits)**

4.4.1 Prefix Codes

- **Prefix Code**
  - **No codeword for one character constitutes the beginning of the codeword for another character.**
  - **We need not look ahead when parsing the file.**
  - **Code 4.2 is an example of a prefix code.**

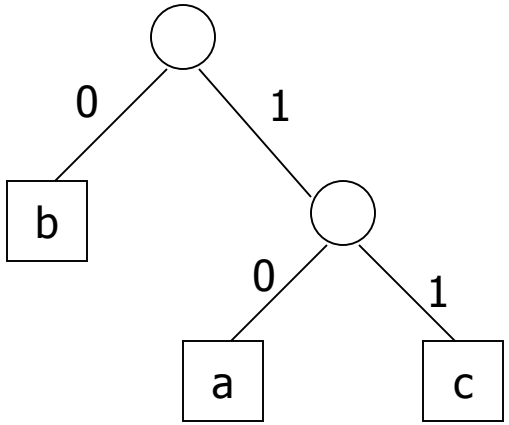


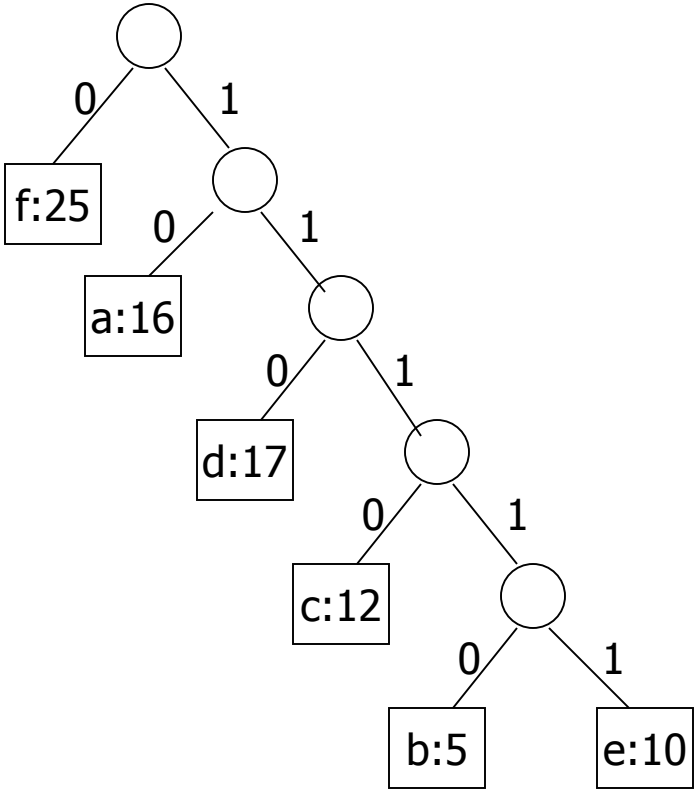
Fig. 4.9 Binary Tree Representation for Code 4.2

4.4.1 Prefix Codes(Example 4.7)

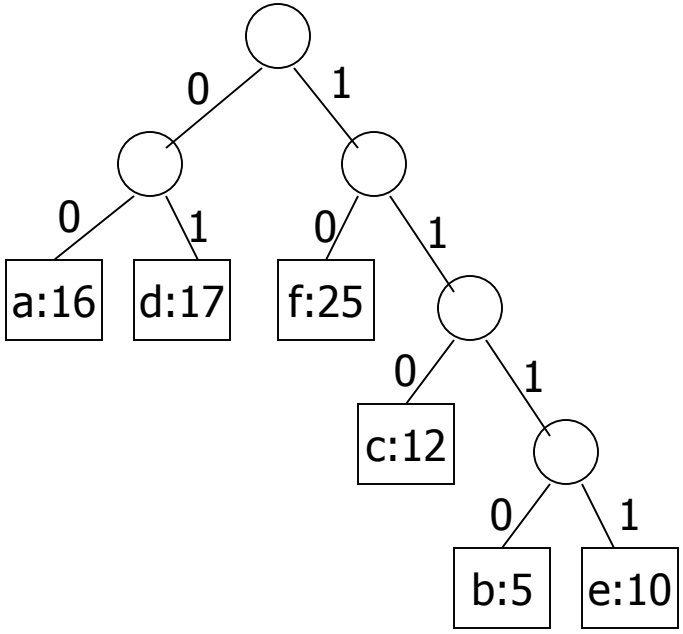
Character	Frequency	C1(Fixed Length)	C2	C3(Huffman)
a	16	000	10	00
b	5	001	11110	1110
c	12	010	1110	110
d	17	011	110	01
e	10	100	11111	1111
f	25	101	0	10

- **Example 4.7: C3 is the best of three**
  - **Bits(C1) = 16(3) + 5(3) + 12(3) + 17(3) + 10(3) + 25(3) = 255**
  - **Bits(C2) = 16(2) + 5(5) + 12(4) + 17(3) + 10(5) + 25(1) = 231**
  - **Bits(C3) = 16(2) + 5(4) + 12(3) + 17(2) + 10(4) + 25(2) = 212**

4.4.1 Prefix Codes(Example 4.7)



Binary Character Code
for Code C2



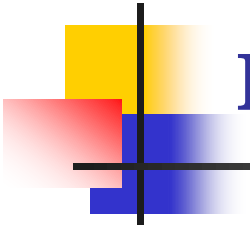
Binary Character Code
For Code C3(Huffman)

4.4.1 Prefix Codes

Given binary tree T

$$bits(T) = \sum_{i=1}^{n} frequency(v_i)depth(v_i)$$

Goal : Find T s.t. bits(T) is min



Priority queue (See Section 7.6)

- **The element with the highest priority is always removed next.**
- **A priority queue can be implemented efficiently as a heap.**
- **A complete binary tree**
  - **All internal nodes have two children**
  - **All leaves have depth *d*.**
- **An essentially complete binary tree**
  - **It is a complete binary tree down to a depth of *d*-1**
  - **The nodes with depth *d* are as far to the left as possible**
- **A heap**
  - **The values stored at the nodes come from an ordered set.**
  - **The value stored at each node is greater than or equal to the values stored at its children. This is called heap property.**

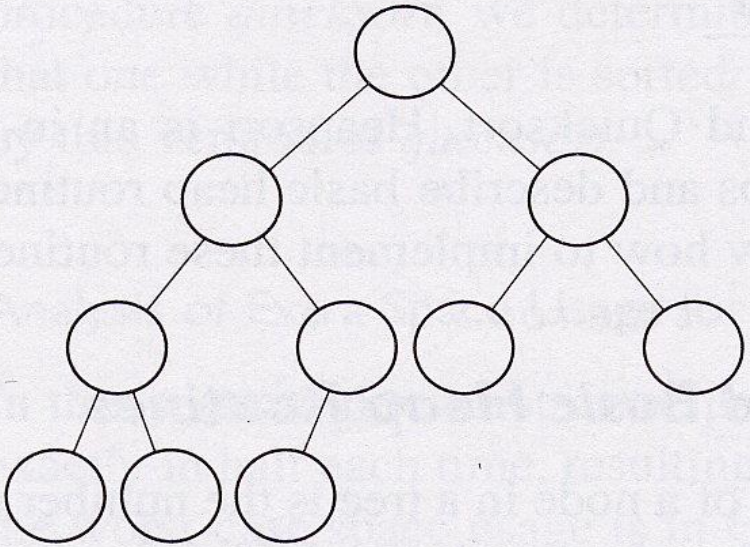


Figure 7.4 An essentially complete binary tree.

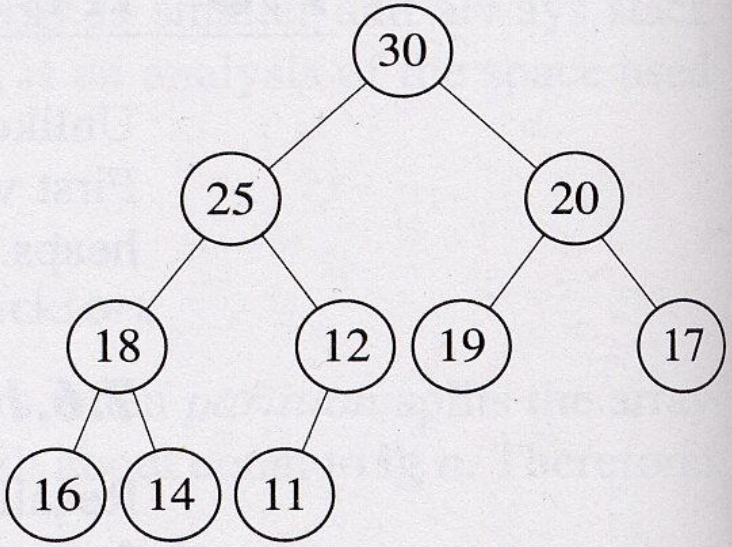
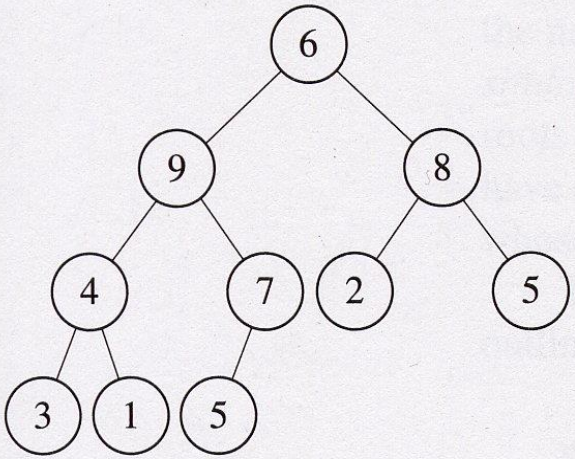


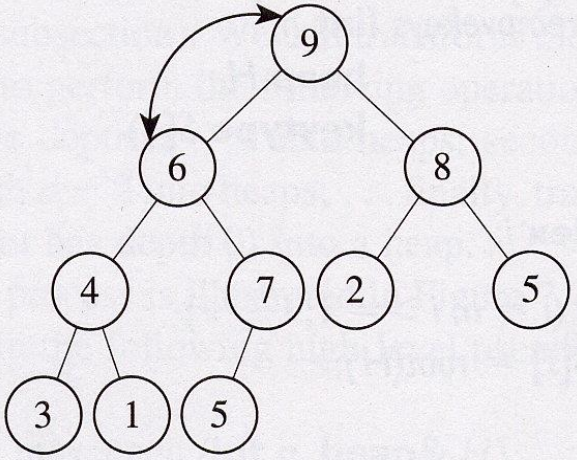
Figure 7.5 A heap.

Figure 7.6 Procedure *siftdown* sifts 6 down until the heap property is restored.

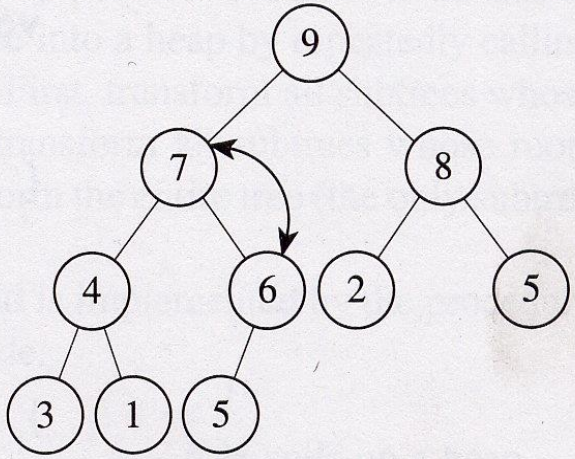
(a) Not a heap



(b) Keys 6 and 9 swapped



(c) Keys 6 and 7 swapped



Heap

Array representation of the heap

30	25	20	18	12	19	17	16	14	11

if index of an element = i

parent = i/2

left child = 2i

right child = 2i+1

4.4.2 Huffman's Algorithm

```
struct nodetype
{
    char  symbol;
    int frequency;
    nodetype* left, right;
}
```

In a priority queue, the element with the highest priority is always removed next.

```
for (i=1; i <= n-1; i++) {          // n = #characters in the file
  remove(PQ, p);  remove(PQ, q);
  r = new nodetype;
  r->left = p;  r->right = q;
  r->frequency = p->frequency + q->frequency;
  insert(PQ, r);
}
remove(PQ, r);
return r;
```

4.4.2 Huffman's Algorithm

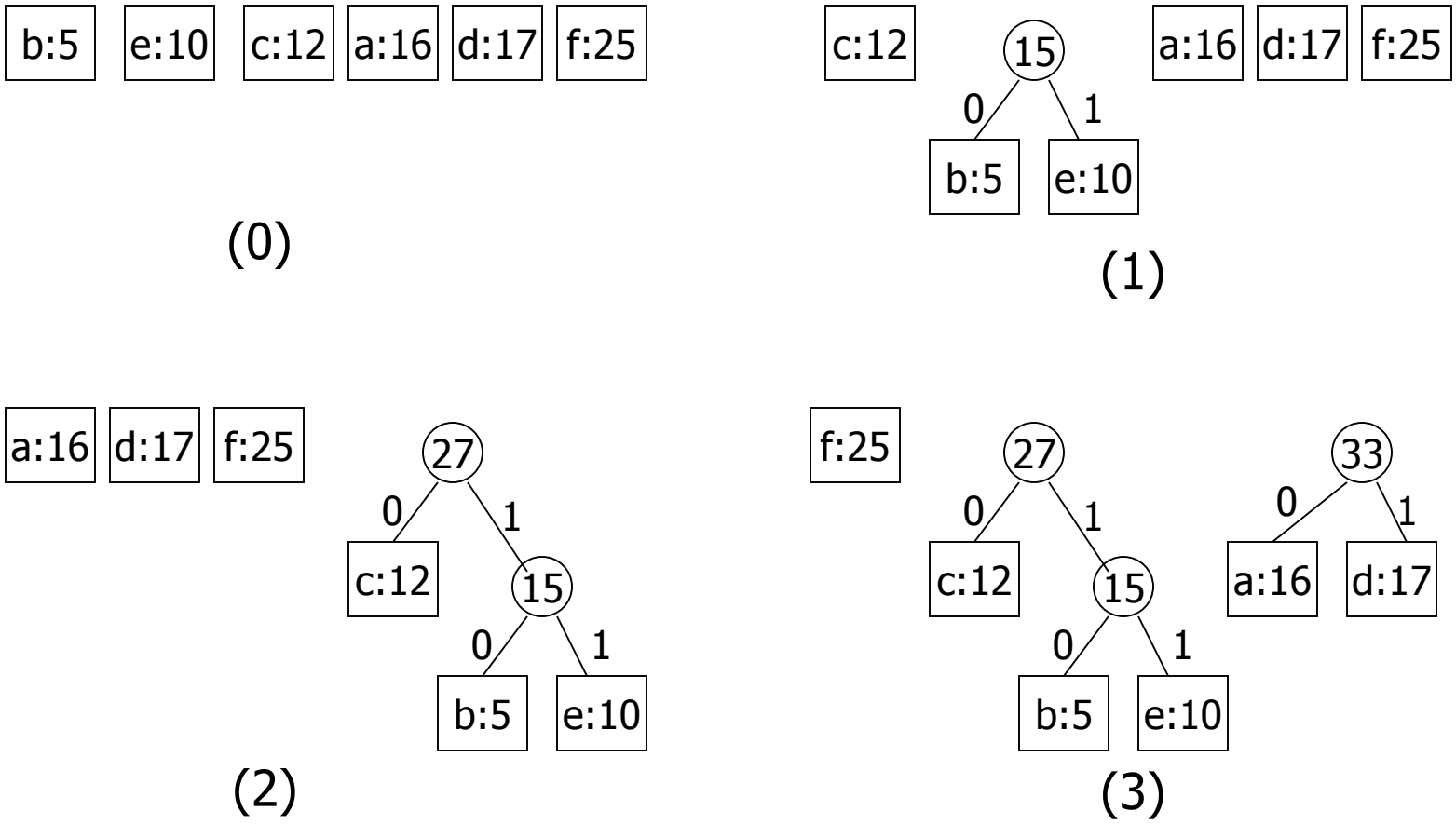


Fig. 4.11 after each pass through the for-*i* loop(1/2)

4.4.2 Huffman's Algorithm

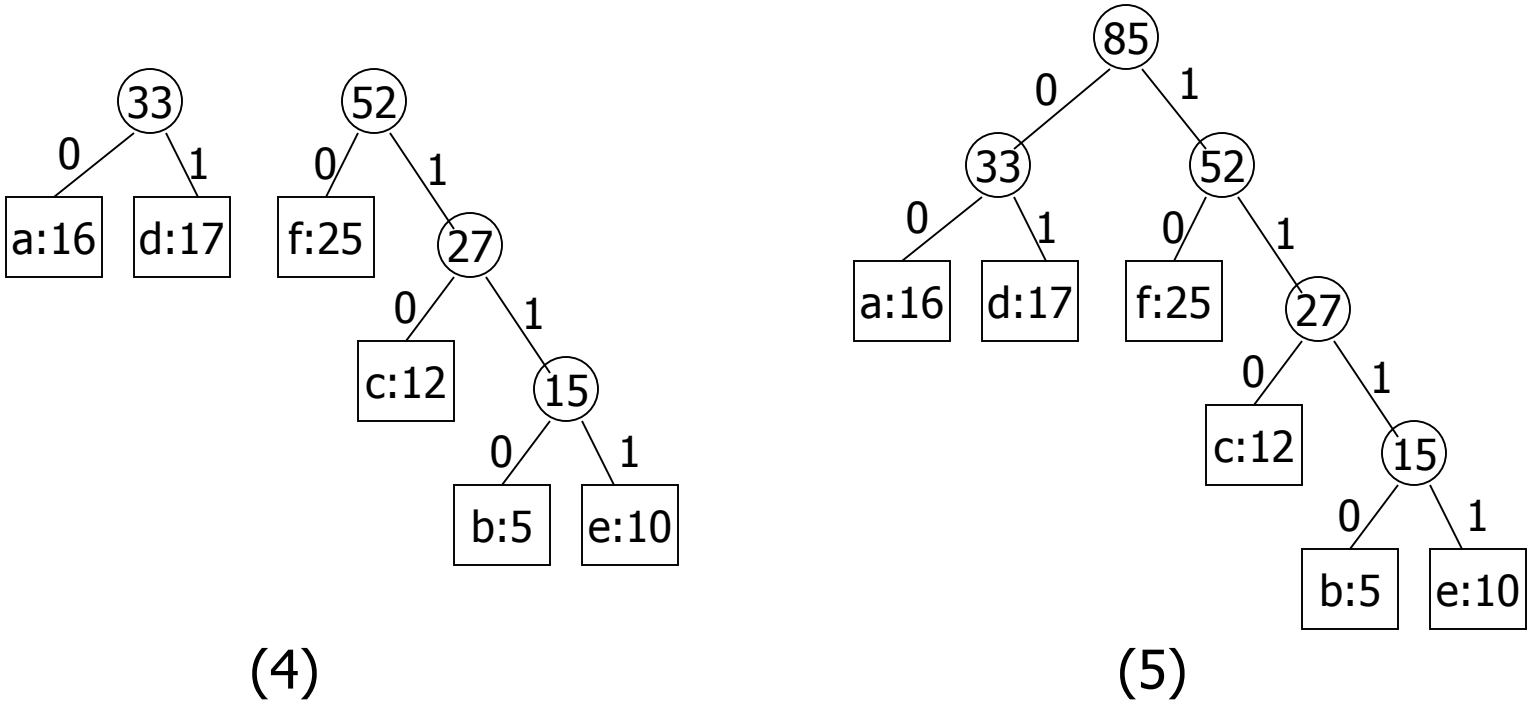


Fig. 4.11 after each pass through the for-*i* loop(2/2)

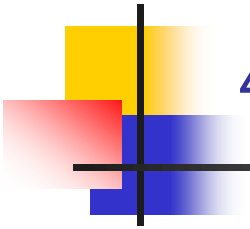
4.4.2 Huffman's Algorithm

[Lemma 4.4] The binary tree corresponding to an optimal binary prefix code is full. i.e., every nonleaf has two children.

Proof: left as an exercise.

Def) A branch with root *v* in tree *T* is the subtree whose root is *v*.

Def) Two nodes are called siblings in a tree if they have the same parent



4.4.2 Huffman's Algorithm

[Theorem 4.5] Huffman's algorithm produces an optimal binary code.

Proof by induction:

- Assuming the set of trees in the i-th step are branches in a binary tree corresponding to an optimal code, we show the set of trees in the (i+1)st step are also branches in the tree.

- We conclude then that (n-1)st step binary tree corresponds to an optimal tree.

4.4.2 Huffman's Algorithm

Induction base: Clearly, the set of single nodes in 0-th step are branches in a tree corresponding to an optimal code.

Induction hypothesis: Assume the set of trees in the i-th step are branches in a tree(say T) corresponding to an optimal code.

Induction step: Let u and v be the roots of the trees combined in the (i+1)st step. If u and v are siblings in T, we are done because the set of trees in (i+1)st step are branches in T.

Otherwise, without loss of generality, assume in T

$$\text{depth}(u) \geq \text{depth}(v).$$

Due to Lemma 4.4, u has some sibling w in T, so

$$\text{depth}(w) \geq \text{depth}(v).$$

Since v was chosen by Huffman's algorithm,

$$\text{frequency}(w) \geq \text{frequency}(v).$$

4.4.2 Huffman's Algorithm

A new binary tree T' is created by swapping the branches v and w in T.

$$\text{bits}(T') = \text{bits}(T) + [\text{depth}(w)-\text{depth}(v)][\text{frequency}(v)-\text{frequency}(w)]$$
$$\leq \text{bits}(T),$$

which means the code corresponding to T' is optimal.

Clearly, the set of trees obtained in the (i+1)st step of Huffman's algorithm are branches in T'.

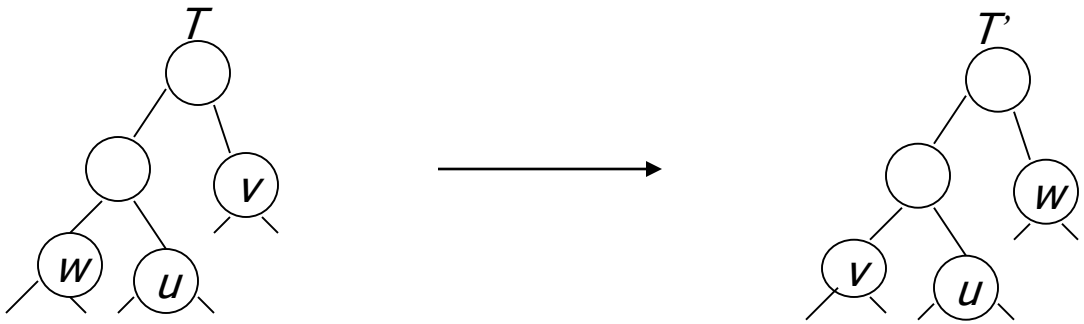


Fig. 4.12 The branches rooted at u and v are swapped.

4.5 The Knapsack Problem

- item i : profit p_i, weight w_i $\quad 1 \le i \le n$

  W : knapsack capacity

  If a fraction x_i $(0 \le x_i \le 1)$ of item i is placed in the knapsack, then a profit of $p_i x_i$ is earned.

- Goal $\quad \max \sum_{i=1}^{n} p_i x_i$

  subject to

  $$\sum_{i=1}^{n} w_i x_i \le W$$

Example

$W = 20 \quad (p_1, p_2, p_3) = (25, 24, 15) \quad (w_1, w_2, w_3) = (18, 15, 10)$

- **Select the items with the largest profit first.**

$$(x_1 \ x_2 \ x_3) = (1, \frac{2}{15}, 0) \quad 25 \times 1 + 24 \times \frac{2}{15} + 15 \times 0 = 28.2$$

- **Select the items with the lightest weight first.**

$$(x_1 \ x_2 \ x_3) = (0, \frac{2}{3}, 1) \quad 25 \times 0 + 24 \times \frac{2}{3} + 15 \times 1 = 31$$

- **Select the items with the largest profit per unit weight first.**

$$(x_1 \ x_2 \ x_3) = (0, 1, \frac{1}{2}) \quad 25 \times 0 + 24 \times 1 + 15 \times \frac{1}{2} = 31.5$$

→ **Greedy approach for Fractional KSP**

0/1 KSP

- $x_i = 0$ or 1
- **GA does not produce an optimal solution**
- **Ex)**

  $W = 30 \quad (p_1, p_2, p_3) = (50, 60, 140) \quad (w_1, w_2, w_3) = (5, 10, 20)$

  - **GA:** $(x_1, x_2, x_3) = (1, 0, 1) \quad 50 + 140 = 190$
  - **Optimal:** $(x_1, x_2, x_3) = (0, 1, 1) \quad 60 + 140 = 200$

4.5.3 DP for 0/1 KSP

- **Define *KNAP*(*s*, *Y*) as follows:**

$$\max \sum_{i=1}^{s} p_i x_i$$

subject to

$$\sum_{i=1}^{s} w_i x_i \le Y$$

$$x_i = 0 \text{ or } 1 \quad 1 \le i \le s$$

item 1 2 ⋯ s ⋯ n

- **Goal: solve *KNAP*(*n*, *W*)**

Principle of optimality

- $y_1, y_2, \cdots, y_n$ **optimal solution for** ***KNAP(n, W)***
- **if** $y_n = 0$, $y_1, \cdots, y_{n-1}$ **is an optimal solution for** ***KNAP(n – 1, W)***

  ex) $W = 10$, $P = (10, 4, 3, 5, 2)$, $W = (5, 4, 2, 3, 2)$

  $Y = (1, 0, 1, 1, 0)$ →

  $(1, 0, 1, 1)$: opt. sol. for $W = 10$, $P = (10, 4, 3, 5)$, $W = (5, 4, 2, 3)$
- **if** $y_n = 1$, $y_1, \cdots, y_{n-1}$ **is an optimal solution for** ***KNAP(n – 1, W – w_n)***

  ex) $W = 10$, $P = (4, 3, 5, 2, 10)$, $W = (4, 2, 3, 2, 5)$

  $Y = (0, 1, 1, 0, 1)$ →

  $(0, 1, 1, 0)$: opt. sol. for $W = 5$, $P = (4, 3, 5, 2)$, $W = (4, 2, 3, 2)$

Generalization (1/2)

- **Let $P[i][x]$ be the optimal profit for $KNAP(i, x)$**

$$P[n][W] = \begin{cases} \max(P[n-1][W],\ p_n + P[n-1][W-w_n]) & \text{if } w_n \le W \\ P[n-1][W] & \text{if } w_n > W \end{cases}$$

$$P[i][x] = \begin{cases} \max(P[i-1][x],\ p_i + P[i-1][x-w_i]) & \text{if } w_i \le x \\ P[i-1][x] & \text{if } w_i > x \end{cases}$$

$P[0][x] = 0$

Want $P[n][W]$

Example 4.9

$W = 30 \quad (p_1, p_2, p_3) = (50, 60, 140) \quad (w_1, w_2, w_3) = (5, 10, 20)$

$P[1][0] = 0$

$P[1][10] = 50$

$P[1][20] = 50$

$P[1][30] = 50$

$$P[2][10] = \begin{cases} \max(P[1][10],\ 60 + P[1][0]) & \text{if } w_2 = 10 \le 10 \\ P[1][10] & \text{if } w_2 = 10 > 10 \end{cases}$$

$$= 60$$

Example 4.9

$$P[2][30] = \begin{cases} \max(P[1][30],\ 60 + P[1][20]) & \text{if } w_2 = 10 \le 30 \\ P[1][30] & \text{if } w_2 = 10 > 30 \end{cases}$$

$$= 60 + 50 = 110$$

$$P[3][30] = \begin{cases} \max(P[2][30],\ 140 + P[2][10]) & \text{if } w_3 = 20 \le 30 \\ P[2][30] & \text{if } w_3 = 20 > 30 \end{cases}$$

$$= 140 + 60 = 200$$

Different Approach

- **Increasingly, the strategy before can be thought as a systematic enumeration of 2^n possibilities for $x_1, x_2, \cdots, x_n$**

  Let S^i represent the possible states reulting from the 2^i decision sequence for $x_1, \cdots, x_i$.

  A state refers to a pair (p, w), w being the total weight of items included in the knapsack and p be the corresponding profit.

  To obtain S^{i+1}

  if $x_{i+1} = 0$ the resulting states are the same as for S^i

  if $x_{i+1} = 1$ the resulting states are obtained by adding (p_{i+1}, w_{i+1}) to each state in S^i

  Let the set of these states be T^{i+1}

  $$T^{i+1} = \{(p + p_{i+1}, w + w_{i+1}) \mid (p, w) \in S^i\}$$

  Then, $S^{i+1} = S^i \cup T^{i+1}$

Example

- n =3, W= 6

$(p_1, p_2, p_3) = (1, 2, 5) \quad (w_1, w_2, w_3) = (2, 3, 4)$

- Solution

$S^0 = \{(0, 0)\}$ +(1, 2) $T^1 = \{(1, 2)\}$

$S^1 = \{(0, 0)\ (1, 2)\}$ +(2, 3) $T^2 = \{(2, 3)\ (3, 5)\}$

$S^2 = \{(0, 0)\ (1, 2)\ (2, 3)\ (3, 5)\}$ +(5, 4) $T^3 = \{(5, 4)\ (6, 6)\ \underbrace{(7, 7)}_{\text{x}}\ \underbrace{(8, 9)}_{\text{x}}\}$

$S^3 = \{(0, 0)\ (1, 2)\ (2, 3)\ \underbrace{(3, 5)}_{\text{x}}\ (5, 4)\ (6, 6)\ \}$

Observations and solution generation

- **Dominance rule**

  (p_k, w_k) dominates (p_j, w_j) if $(p_j \le p_k)$ and $(w_j \ge w_k)$
  ex) (5, 4) dominates (3, 5)

- **Purge (p, w) with $w > W$**

  ex) purge (7, 7) (8, 9)

- **Determination of x_i**

  If (p, w) is the last tuple in S^n, we can set

  $x_n = 0$ if $(p,w) \in S^{n-1}$

  $x_n = 1$ if $(p,w) \notin S^{n-1}$ /* $(p - p_n, w - w_n) \in S^{n-1}$ */

  Determine the remain x_i recursively

  ex) $(6,6) \in S^3$ $(6,6) \notin S^2$ $x_3 = 1$

  $(6 - p_3, 6 - w_3) = (1,2) \in S^2$ $(1,2) \in S^1$ $x_2 = 0$

  $(1,2) \in S^1$ $(1,2) \notin S^0$ $x_1 = 1$

T(n) and M(n)

$$|T^i| \le |S^{i-1}| \qquad |S^i| \le 2|S^{i-1}|$$

$$\sum_{i=0}^{n-1} |S^i| = \sum_{i=0}^{n-1} 2^i = 2^n - 1 \in \Theta(2^n)$$

- **Special case**

If p_j's are integer $\quad |S^i| \le 1 + \sum_{j=1}^{i} p_j$

If w_j's are integer $\quad |S^i| \le 1 + \min\{ \sum_{j=1}^{i} w_j, W \}$

$$T(n) \in \Theta(\min\{ 2^n, n \sum_{i=1}^{n} p_i, nW \})$$