5 最短路問題とダイクストラ法

カーナビゲーションやポータルサイトの路線情報は，出発地から目的地までの最適な経路を線形計画問法で求めています．**最短路問題** (shortest-path problem) とよばれるこの LP 問題は，その特殊な問題構造から一般の LP よりも易しく，シンプレックス法とは異なる専用の効率的なアルゴリズムも開発されています．アルゴリズムの仕組みを見るまえに，最短路問題の構造を与える**グラフ** (graph) について簡単に説明しておきましょう．

5.1 グラフ

有限個の**頂点** (vertex, node) の集合 $V = \{1, 2, \ldots, m\}$ と，頂点対の集合 $E \subseteq V \times V \equiv \{(i, j) \mid i \in V,\ j \in V\}$ の組をグラフといい，$G = (V, E)$ で表します．集合 E に属する頂点対 $e = (i, j)$ をグラフ G の**枝** (arc, edge)，頂点 i と j を枝 e の**端点** (end node) とよび，枝 e は頂点 i, j に**接続する**といいます．どの枝の向きも考えないときは G を**無向グラフ** (undirected graph)，枝の向きを考えて (i, j) と (j, i) を区別するときには**有向グラフ**とよびます．有向グラフでは枝 $e = (i, j)$ の頂点 i, j をそれぞれ e の**始点** (tail)，**終点** (head) といいます．

グラフ $G = (V, E)$ に対して V, E の部分集合をそれぞれ V', E' とするとき，任意の $e' \in E'$ の両端点が V' に属すならば，$G' = (V', E')$ は再びグラフとなります．そのような G' を G の**部分グラフ** (subgraph) といいます．特に $G = (V, E)$ で $V' \subseteq V$ に両端点をもつ枝の集合を E' とするとき，$G' = (V', E')$ は V' による G の**生成部分グラフ** (induced subgraph) であるといいます．

路. 有向グラフ $G = (V, E)$ の頂点の列 $P = (i_0, i_1, \ldots, i_p)$ が $(i_k, i_{k+1}) \in E$ $(k = 0, 1, \ldots, p-1)$ を満たすとき，P を頂点 i_0 から i_p への**有向路** (directed path) とよびます．**無向路** (undirected path) も同様に定義されますが，無向路 $P = (i_0, i_1, \ldots, i_p)$ では隣接する 2 つの頂点 i_k, i_{k+1} に対して $(i_k, i_{k+1}) \in E$ か $(i_{k+1}, i_k) \in E$ のいずれか $(k = 0, 1, \ldots, p-1)$ が満たされます．有向路と無向路をあわせて単に**路** (path) とよびますが，路 P を頂点あるいは枝の集合とみなして $i \in P$ や $(i, j) \in P$ などの表現を用います．

始点 i_0 と終点 i_p が同じ頂点である路を**閉路** (circuit)，含まれる頂点 $i_0, i_1, \ldots, i_p$ がすべて異なる路を**単純路** (simple path)，その両方の性質をもつ路を**単純閉路** (simple circuit) とよびます．これらの閉路，単純路に対しても有向，無向が定義されます．

連結． グラフ G の任意の 2 つの頂点間に無向路が存在するとき，G は**連結グラフ** (connected graph) であるといいます．グラフ G が連結でなくとも，その連結な生成部分グラフ G' で，G' を真に含む連結生成部分グラフが存在しないとき，G' を G の**連結成分** (connected component) といいます．連結でないグラフは互いに頂点を共有しないいくつかの連結成分に分解されます．

木． グラフ G が閉路を含まないとき，G を**非巡回的** (acyclic) であるといいます．また，非巡回的な連結グラフ T を**木** (tree) とよびます．頂点数 n の連結グラフ T が木であるための必要十分条件は，T が $n-1$ 本の枝をもつことです．次数が 1 である木の頂点を**葉** (leaf) とよびますが，木には少なくとも 2 枚の葉が存在します．グラフ $G=(V,E)$ に対して G と同じ頂点集合をもつ部分グラフ $G'=(V,E')$ が木であるとき，G' を G の**全域木** (spanning tree) といいます．

5.2 最短路問題

グラフ $G=(V,E)$ と各枝 $(i,j)\in E$ の長さ c_{ij} によって定まるネットワーク上で特定の頂点 s から他のすべての頂点への最短路を求める問題は次のように線形計画問題として定式化できます:

$$\left|\begin{array}{ll} \text{最小化} & \displaystyle\sum_{(i,j)\in E} c_{ij}x_{ij} \\ \text{条　件} & \displaystyle\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = \begin{cases} n-1, & i=s \\ -1, & \forall i\in V\setminus\{s\} \end{cases} \\ & 0\le x_{ij}\le n, \quad \forall (i,j)\in E. \end{array}\right. \tag{5.1}$$

以下では，

仮定 5.1. グラフ G には頂点 s から各頂点 $i\in V\setminus\{s\}$ への有向路が存在する

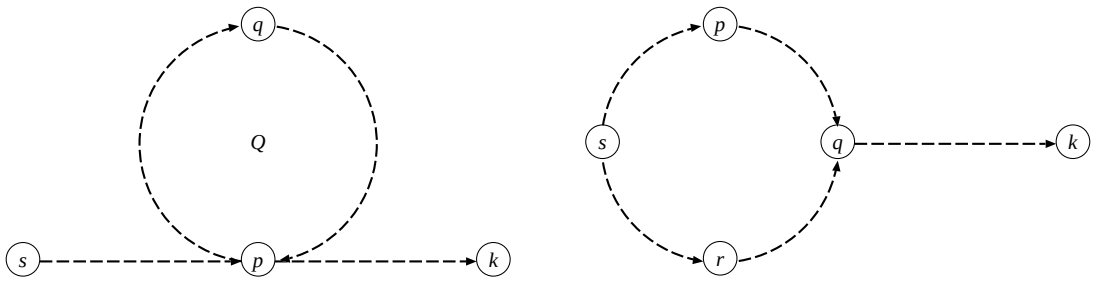

図 5.1: 性質 5.1 と性質 5.2.

ものとします．もしも頂点 s から i への有向路が存在しなければ，人工的に枝 (s,i) を G に加え，その長さを $c_{si}=+\infty$ と定義することで仮定 5.1 は一般性を失いません．

アルゴリズムを紹介する前に問題 (5.1) の基本的な性質を少し調べておきましょう．

性質 5.1. 頂点 s から k への最短路が存在すれば，頂点 k への最短の単純有向路が存在する．

性質 5.2. 有向路 $P_k=(s=i_1,i_2,\ldots,i_h=k)$ が頂点 k への最短路ならば，各 $\ell=2,3,\ldots,h-1$ に対して $P_\ell=(i_1,\ldots,i_\ell)$ は頂点 i_ℓ への最短路である．

性質 5.3. 任意の頂点 $i\in V$ への最短距離を $d(i)$ で表すとき，有向路 P_k が頂点 k への最短路であるための必要十分条件は，

$$d(j)=d(i)+c_{ij},\quad \forall(i,j)\in P_k. \tag{5.2}$$

図 5.1 の左の絵を使って性質 5.1 を示しましょう．これは，頂点 k への有向路を表していますが，そこには有向閉路 Q が含まれています．もし，

$$\sum_{(i,j)\in Q} c_{ij}<0 \tag{5.3}$$

が成り立たなければ，閉路 Q を通る価値はなく，頂点 p から直接 k へ向かう方が距離は短くなります．では，(5.3) が成り立てばどうでしょうか？ 有向閉路 Q を 1 周するごとに頂点 k への距離は短くなるので，頂点 k への最短路を定めることはできません．

次の性質 5.2 も容易に示すことができます．図 5.1 の右の絵において，頂点 k への最短路を $P_k=(s,\ldots,p,\ldots,q,\ldots,k)$ とし，部分路 $P_q=(s,\ldots,p,\ldots,q)$ は q への最短路になっていな

いものと仮定しましょう．すると，

$$\sum_{(i,j)\in P} c_{ij} < \sum_{(i,j)\in P_q} c_{ij}$$

を満たす頂点 s から q への最短路 $P = (s, \ldots, r, \ldots, q)$ が存在します．したがって，P_q の代わりに P を通る頂点 k への有向路 $P'_k = (s, \ldots, r, \ldots, q, \ldots, k)$ は

$$\sum_{(i,j)\in P'_k} c_{ij} < \sum_{(i,j)\in P_k} c_{ij}$$

となって，P_k が最短であることに矛盾します．

性質 5.3 の (5.2) が必要条件であることは性質 5.2 から直ちに導かれます．十分性を示しましょう:

いま，$P_k = (s = i_1, \ldots, i_h = k)$ とすれば，$d(i_1) = 0$ なので

$$d(k) = d(i_h) = (d(i_h) - d(i_{h-1})) + (d(i_{h-1}) - d(i_{h-2})) + \cdots + (d(i_2) - d(i_1))$$

が成り立ちます．もしも (5.2) が満たされれば，各枝 $(i, j) \in P_k$ に対して $d(j) - d(i) = c_{ij}$ となり，

$$d(k) = c_{i_{h-1}i_h} + c_{i_{h-2}i_{h-1}} + \cdots + c_{i_1 i_2} = \sum_{(i,j)\in P_k} c_{ij}$$

が得られます．

性質 5.4. すべての頂点 k への最短路によってグラフ G の全域木を構成できる．

性質 5.1 により，以後，各頂点 k への最短路 P_k には閉路が含まれていないものと仮定することにします．さて，P_K を枝の集合と考え，

$$T = \bigcup_{k\in V} P_k \subseteq E$$

を定義しましょう．枝集合 T は各最短路 P_k をじょうずに選択することで（P_k を単純路と仮定しても一意とは限らないことに注意）グラフ G の全域木をつくることができます．

仮に T が全域木でないとすると，各最短路には閉路が含まれないことから，図 5.2 に示すように複数の最短路によって閉路 Q が構成されるはずです．頂点 k, ℓ への最短路をそれぞれ

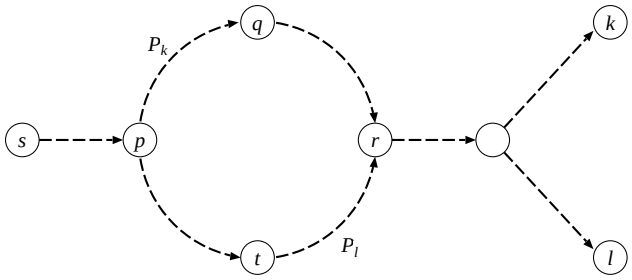

図 5.2: 性質 5.4.

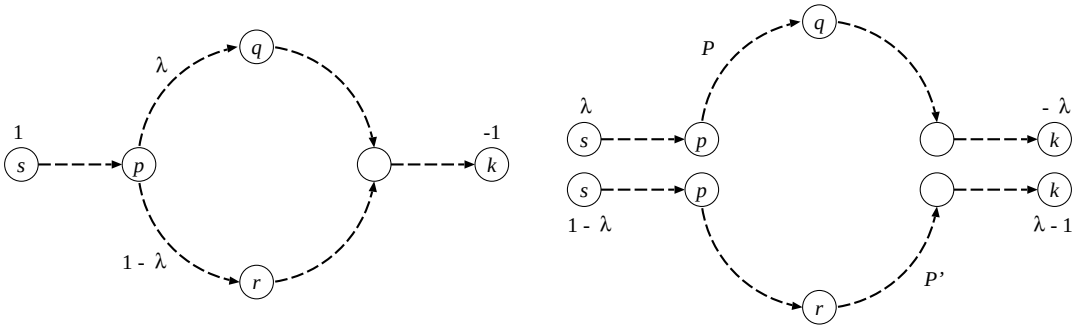

図 5.3: 性質 5.5.

$P_k = (s, \ldots, p, \ldots, q, \ldots, r, \ldots, k)$, $P_\ell = (s, \ldots, p, \ldots, t, \ldots, r, \ldots, \ell)$ とすれば，性質 5.2 により，その部分路 $P_r = (s, \ldots, p, \ldots, q, \ldots, r)$ と $P' = (s, \ldots, p, \ldots, t, \ldots, r)$ の長さはともに $d(r)$ に等しくなります．したがって，P_r を P'_r と入れ替えた $P'_k = (s, \ldots, p, \ldots, t, \ldots, r, \ldots, k)$ も最短路となり，$T = (T \setminus P_k) \cup P'_k$ とすることで閉路 Q は解消されます．

性質 5.5. 問題 (5.1) に最適流が存在すれば，整数ベクトルとなる最適流 $\mathbf{x}^*$ が存在する．

各頂点 k への最短路 P_k を使って問題 (5.1) の最適流 $\mathbf{x}^*$ を表してみましょう．最短路 P_k は単純路なので，各枝 $(i,j) \in E$ を高々 1 度しか通りません．そこで，

$$\delta(P_k) = \begin{cases} 1, & (i,j) \in P_k \text{ のとき} \\ 0, & (i,j) \notin P_k \text{ のとき}, \end{cases}$$

と定めれば，$\mathbf{x}^*$ は

$$x^*_{ij} = \sum_{k \in V} \delta(P_k) \tag{5.4}$$

によって与えられることがわかります．逆に，各頂点 k への最短路 P_k を与える最適流は本当に存在するのでしょうか？

問題 (5.1) では，x_{ij} が実数値をとることも許されています．仮に整数値をとらない x^*_{ij} が存在するとすれば，図 5.3 の左に示すようにある需要点 k は供給点 s から 1 単位の流れを複数の経路を通して受け取るはずです．この例の場合，右図のように頂点 k へは 2 つの有向路 $P = (s, \ldots, p, \ldots, q, \ldots, k)$, $P' = (s, \ldots, p, \ldots, r, \ldots, k)$ が存在し，それぞれに $\lambda, 1-\lambda\ (>0)$ 単位が流れることになります．このとき，頂点 s から k へ 1 単位の流れを送るのに必要な費用は

$$c(\lambda) = \lambda \sum_{(i,j)\in P} c_{ij} + (1-\lambda) \sum_{(i,j)\in P'} c_{ij}$$

です．しかし，

$$c(\lambda) \geq \min \left\{ \sum_{(i,j)\in P} c_{ij}, \sum_{(i,j)\in P'} c_{ij} \right\}, \quad 0 < \forall\lambda < 1,$$

が成り立つことから， 2 つの有向路 P, P' のうち一方にだけ 1 単位を流しても費用は増加しません; その一方の有向路が頂点 k への最短路です．この観察からわかるのは: 整数，実数を問わず，問題 5.1 に最適流が存在すれば，頂点 s から各頂点 k への最小費用の有向路，つまり最短路 P_k が存在する; そして，そこに 1 単位を流すことで得られる整数解も (5.1) の最適解となる，ことです.

5.3 ダイクストラ法

ダイクストラ法 (Dijkstra's algorithm) は最短路問題のアルゴリズムの中で理論的に最も効率的なことが知られていますが，枝の長さが

仮定 5.2.
$$c_{ij} \geq 0, \quad \forall (i,j) \in E$$

を満たさなければ正しく機能しません．基本となる操作は，出発点 s から扇状に探索の木を広げていき，頂点 s から近い順に各頂点 i への距離の記されたラベル $d(i)$ を貼りつけることです．この距離ラベルには，本当の最短距離の記された**永久ラベル** (permanent label) と，仮

の最短距離の記された**一時ラベル** (temporary label) があり，頂点集合 V は前者の貼られた集合 $\mathcal{P}$ と後者の貼られた集合 $\mathcal{T}$ に分割されます．

アルゴリズムはまず，$\mathcal{P} = \{s\}$, $\mathcal{T} = V \setminus \{s\}$ として，頂点 s には永久ラベル $d(s) = 0$ を貼り，他の頂点 j には一時ラベル

$$d(j) = \begin{cases} c_{sj}, & (s,j) \in E \text{ のとき} \\ \infty, & \text{そうでないとき}, \end{cases}$$

を貼ります．各反復では，永久，一時にかかわらず，どのラベル $d(j)$ も，

(a) 出発点 s から途中，**集合 $\mathcal{P}$ の頂点だけを通って** j へ向かう有向路の中で最短の長さ

を表します．アルゴリズムは，一時ラベルの頂点集合 $\mathcal{T}$ の中から最も小さい距離ラベル $d(i)$ の頂点 i を選んで $d(i)$ を永久ラベルとし，i を始点とする枝の集合:

$$E(i) = \{(i,j) \in E \mid j \in V\}$$

の各終点のラベルが (a) を保つように更新します．すべての頂点に永久ラベルが貼られた時点，つまり $\mathcal{P} = V$ ととなったら終了です．

algorithm DIJKSTRA

```
begin
  P := {s}; T := V \ {s};
  d(s) := 0; pred(s) := 0;
  if (s,j) ∈ E then d(j) := c_sj と pred(j) := s を初期化する
  else d(j) := ∞;
  while P ≠ V do begin
    d(i) = min{d(j) | j ∈ T} を満たす頂点 i を選ぶ;          { 頂点の選択 }
    P := P ∪ {i}; T := T \ {i};
    for 各 (i,j) ∈ E(i) do                                  { ラベルの更新 }
      if d(j) > d(i) + c_ij then d(j) := d(i) + c_ij と pred(j) := i を更新する
  end
end;
```

各反復で pred(i) は，長さ $d(i)$ の有向路で頂点 i の直前に訪れる頂点を指し示すように更新されます．したがって終了時には，この指標をもとに各頂点 i から出発点 s までの最短路を逆に辿ることができます．

定理 5.6.　DIJKSTRA はグラフ $G = (V, E)$ の 1 つの頂点 $s \in V$ から他のすべての頂点 $i \in V \setminus \{s\}$ への最短路を与える．

証明:　帰納法によって証明しましょう．

ある反復で，各ラベル $d(j)$ は (a) を満たし，それが永久ラベル，つまり $j \in \mathcal{P}$ ならば，

(b)　出発点 s から j へ向かう本当の最短路の長さ

を表しているものと仮定する．次の反復では，

$$d(i) = \min\{d(j) \mid j \in \mathcal{T}\} \tag{5.5}$$

を満たす頂点 i が選ばれ，$\mathcal{P} := \mathcal{P} \cup \{i\}$ と更新されるが，この頂点 i のラベル $d(i)$ が「出発点 s からの本当の最短距離」となっていることを示せば，再び (a), (b) が満足されて証明終了である．これには，出発点 s から i への任意の有向路 P の長さが $d(i)$ 以上となることを示せばよい．

有向路 P を頂点 $s \in \mathcal{P}$ から辿ったとき，最初に訪れる $\mathcal{T}$ の頂点を k とすれば，(5.5) より，

$$d(k) \geq d(i)$$

である．一方，仮定 5.2 より，頂点 s から i までの P に含まれる枝の長さはすべて非負である．この 2 つから，有向路 P の長さは $d(i)$ より短くなりえないことがわかる． □

宿題

5.1　常磐線の登だけを使い，柏か松戸のいずれかで必ず下車して土浦から上野まで向かうとき，最も安い旅程を提案しなさい．

6 ナップサック問題と動的計画法

最適化問題の中には，前節の最短路問題などよりずっと単純であるにもかかわらず，はるかに難しい問題が数多く存在します．これから紹介する**ナップサック問題** (knapsack problem) はその代表例で，未だにこの問題を上手く解く厳密なアルゴリズムは発見されていません．そのため，最適解を求めるには一種の列挙法によって実行可能解をすべて調べるほか手立てがありません．ここでは，効率よく列挙を行って最適解を見つける**動的計画法** (dynamic programming) を紹介します．

6.1 0-1 ナップサック問題

例 6.1. 品物 G_1, G_2, G_3, G_4 をナップサックに詰めてハイキングに出かけたい．品物の体積の総和がナップサックの容積を越えるとき，どの品物をナップサックに詰めればよいか ？

ナップサックの容積を $b = 12$ (ℓ)，4 つの品物それぞれの体積を $a_1 = 3$, $a_2 = 6$, $a_3 = 4$, $a_4 = 5$ (ℓ) としましょう．この場合，品物の体積は $\sum_{j=1}^{4} a_j = 18$ (ℓ) となってナップサックに入りません．しかし，このハイカーにとっての各品物の価値が $c_1 = 7$, $c_2 = 9$, $c_3 = 5$, $c_4 = 5$ のように数値化されていれば，この問題は最適化問題として定式化することができます．

一般に，品物が $G_1, G_2, \ldots, G_n$ の n 個の場合，それぞれに変数 $x_1, x_2, \ldots, x_n$ を設定し，

$$x_j = \begin{cases} 1, & \text{品物 } G_j \text{ をナップサックに詰めるとき} \\ 0, & \text{品物 } G_j \text{ をナップサックに詰めないとき} \end{cases}$$

と定めます．これにより，問題は次のように定式化できます:

$$\left| \begin{array}{llllllll} \text{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n \\ \text{条　件} & a_1x_1 & + & a_2x_2 & + & \cdots & + & a_nx_n \quad \leq \quad b \\ & & & x_j \in \{0,\ 1\}, & & & & j = 1,\ 2,\ \ldots,\ n \end{array} \right. \tag{6.6}$$

ただし，係数 $a_1, \ldots, a_n, b, c_1, \ldots, c_n$ はすべて正で，

$$a_1 + a_2 + \cdots + a_n > b \tag{6.7}$$

であるものとします．問題 (6.6) は一見，単純な LP にも見えますが，各変数 x_j が 0 または 1 という整数値しか取れない点で LP とは大きく異なります．この種の問題を **0-1 整数計画問題** (0-1 integer programming problem) とよび，一般には

$$\left|\begin{array}{lllllllll} \text{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & \\ \text{条　件} & a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{in}x_n & \leq \quad b, \quad i=1,\,2,\,\ldots,\,m \\ & & & & & & & x_j \in \{0,\,1\}, & \quad j=1,\,2,\,\ldots,\,n \end{array}\right. \tag{6.8}$$

のように定式化され，係数が正であることは必ずしも仮定されません．

問題 (6.6) のもう一つの特徴は，一般の 0-1 計画問題 (6.14) と比較すればわかるように，0-1 条件 $x_j \in \{0,\,1\}$, $j=1,\,2,\,\ldots,\,n$ 以外の制約条件が 1 本しか存在しないことです．そのため，0-1 条件を

$$0 \leq x_j \leq 1, \quad j=1,\,2,\,\ldots,\,n$$

に**連続緩和** (continuous relaxation) すれば，(6.6) からは線形計画問題

$$\left|\begin{array}{lllllllll} \text{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & \\ \text{条　件} & a_1x_1 & + & a_2x_2 & + & \cdots & + & a_nx_n & \leq \quad b \\ & & & 0 \leq x_j \leq 1, & & & & & \quad j=1,\,2,\,\ldots,\,n \end{array}\right. \tag{6.9}$$

が導かれます．いま，c_j/a_j が

$$c_1/a_1 \geq c_2/a_2 \geq \cdots \geq c_n/a_n \tag{6.10}$$

の順になっているものとしましょう (必要ならば，ソートして添字を付け替えます)．品物を G_1 から G_2, G_3, ... の順にナップサックに詰めていき，スペースがあって丸ごと入れば，それを 1 個いれます．もしも最後に入れようとする品物 G_p がスペース不足で丸ごと入らなければ，その一部 (スペース分) だけを入れます．こうして得られる解

$$\overline{x}_j = \begin{cases} 1, & j=1,\,\ldots,\,p-1 \\ \left(b - \displaystyle\sum_{i=1}^{p-1} a_i\right)/a_p, & j=p \\ 0, & j=p+1,\,\ldots,\,n \end{cases}$$

が，実は (6.6) の連続緩和問題 (6.9) の最適解です．

連続緩和問題の最適解 $\overline{\mathbf{x}} = (\overline{x}_1,\ \ldots,\ \overline{x}_n)$ と元の 0-1 整数計画問題の最適解 $\mathbf{x}^* = (x_1^*,\ \ldots,\ x_n^*)$ との間には，品物の詰め方からもわかるように

$$\sum_{j=1}^{n} c_j \overline{x}_j \geq \sum_{j=1}^{n} c_j x_j^* \tag{6.11}$$

の関係があります．しかし，最後の品物 G_p がちょうど 1 個入るとは限らないので，$\overline{\mathbf{x}}$ は必ずしも (6.6) の最適解とはなりません．

6.2 動的計画法

0-1 ナップサック問題 (6.6) の係数 $a_1, \cdots, a_n, b$ は整数であるものと仮定しましょう．計算機上では，すべての数が有理数として扱われるので，この仮定によって不都合は生じません．

ここでは次のような関数を考えます：

$V_k(y)$ ： 容積 y の中に詰める品物の候補を $G_1, \ldots, G_k$ に限定したときに得られる最大の価値．

したがって，(6.6) の最適解は目的関数値が $V_n(b)$ の実行可能解ということになります．この値を求めるため，$V_k(y)$ が満たす次の関係に注目しましょう：

$$V_k(y) = \max\{V_{k-1}(y),\ V_{k-1}(y - a_k) + c_k\} \tag{6.12}$$

ここでは $k = 1,\ 2,\ \ldots,\ n$; $y = 0,\ 1,\ \ldots,\ b$ ですが，

$$\left.\begin{array}{ll} V_0(y) = 0, & y \geq 0 \text{ の場合} \\ V_k(y) = -\infty, & y < 0 \text{ の場合} \end{array}\right\} \tag{6.13}$$

とします．式 (6.12) を**動的計画法** (dynamic programming: DP) の**漸化式** (recursion fomula) とよびますが，「容積 y の中に詰める候補を $G_1, \ldots, G_k$ とするときの最大価値 $V_k(y)$ は，

- y の中に詰める候補を $G_1, \ldots, G_{k-1}$ とするときの最大価値 $V_{k-1}(y)$，
- y の中に，まず G_k を詰め，残った容積 $y - a_k$ に詰める候補を $G_1, \ldots, G_{k-1}$ とするときの最大価値 $V_{k-1}(y - a_k) + c_k$

のどちらか大きいほうに等しい」ことを意味します.

例 6.2. 次の例題を動的計画法を使って実際に解いてみましょう：

$$\left|\begin{array}{lllllllllll} \text{最大化} & 7x_1 & + & 9x_2 & + & 5x_3 & + & 5x_4 & & & \\ \text{条　件} & 3x_1 & + & 6x_2 & + & 4x_3 & + & 5x_n & \leq & 12 \\ & & & x_j \in \{0,\ 1\}, & & & & j = 1,\ 2,\ 3,\ 4. & & \end{array}\right.$$

まず，$V_1(1) = 0$, $V_1(2) = 0$, $V_1(3) = \cdots = V_1(12) = 7$ は明らかでしょう．それぞれに対応する x_1 の値を付けて下のような表を作ります：

k / y	1	2	3	4	5	6	7	8	9	10	11	12
$V_1(y)$	0	0	7	7	7	7	7	7	7	7	7	7
x_1	0	0	1	1	1	1	1	1	1	1	1	1

次に $a_2 = 6$ ですから，

$$V_2(y) = V_1(y), \qquad y \leq 5 \text{ のとき}$$
$$V_2(6) = \max\{V_1(6),\ V_1(6-6) + c_2\} = \max\{7,\ 9\} = 9$$
$$V_2(7) = \max\{7,\ 9\} = 9, \quad V_2(8) = \max\{7,\ 9\} = 9$$
$$V_2(9) = \max\{V_1(9),\ V_1(9-6) + c_2\} = \max\{7,\ 7+9\} = 16.$$

以下同様にして $V_4(12)$ まで計算し，それと同時に max を与える x_j の値も求めます：

k / y	1	2	3	4	5	6	7	8	9	10	11	12
$V_2(y)$	0	0	7	7	7	9	9	9	16	16	16	16
x_2	0	0	0	0	0	1	1	1	1	1	1	1
$V_3(y)$	0	0	7	7	7	9	12	12	16	16	16	16
x_2	0	0	0	0	0	0	1	1	0	0	0	0
$V_2(y)$	0	0	7	7	7	9	9	9	16	16	16	17
x_2	0	0	0	0	0	0	0	0	0	0	0	1

さて，最適解 $\mathbf{x}^* = (x_1^*,\ x_2^*,\ x_3^*,\ x_4^*)$ ですが，次の手順で求めます： まず，

$$V_4(12) = 17, \qquad x_4^* = 1.$$

ここから逆向きに計算していき，

$$\begin{aligned} V_3(12 - a_4) = V_3(7) &= 12, \quad x_3^* = 1 \\ V_2(7 - a_3) = V_2(3) &= 7, \quad x_2^* = 0 \\ V_1(3) &= 7, \quad x_1^* = 1 \end{aligned}$$

となります. □

ここで取りあげた 0-1 ナップサック問題では品物 G_j がそれぞれ 1 個しか用意されていないとしましたが，いくつも用意されている場合も同様に考えることができます．これを定式化すれば，

$$\left| \begin{array}{llllllll} \text{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n \\ \text{条　件} & a_1x_1 & + & a_2x_2 & + & \cdots & + & a_nx_n \quad \leq \quad b \\ & & & x_j \in \mathbb{Z}_+, & & & & j = 1,\ 2,\ \ldots,\ n \end{array} \right. \tag{6.14}$$

となりますが，ここで $\mathbb{Z}_+$ は非負の整数全体の集合を表し，また係数はすべて正で条件 (6.7) を満たすものと仮定します．問題 (6.14) は「0-1」を付けずに単に**ナップサック問題**とよばれますが，これも漸化式

$$V_k(y) = \max\{V_{k-1}(y),\ V_k(y - a_k) + c_k\} \tag{6.15}$$

を用いて動的計画法で解くことができます (なぜか？ 宿題 8.2).

宿題

6.1 0-1 ナップサック問題 (6.6) の最適解 $\mathbf{x}^*$ と，その連続緩和問題の最適解 $\overline{\mathbf{x}}$ の間に (6.11) の関係が成り立つことを，２つの問題の実行可能領域の包含関係に着目して証明しなさい.

6.2 容積 y のナップサックに品物 $G_1, \ldots, G_k$ を詰める場合の価値の総和の最大値を $f(k,\ y)$ で表したとき，

$$f(k,\ y) = \max\{f(k-1,\ y),\ f(k-1,\ y - a_k) + c_k\} \tag{6.16}$$

の関係が成り立つ．0-1 ナップサック問題の最適値は，

$$\begin{cases} f(k,\, y) = -\infty, & y < 0 \text{ のとき} \\ f(0,\, y) = 0, & y \geq 0 \text{ のとき}, \end{cases}$$

と定めて動的計画法を使えば，$f(n,\, b)$ によって与えられる．

(a) 関係式 (6.16) を説明しなさい．

(b) 動的計画法を使って例 6.1 の問題を解きなさい．

6.3 0-1 ナップサック問題では各品物 G_j は 1 つしか存在しなかったが，複数存在する問題を単に**ナップサック問題**とよび，以下のように定式化される:

$$\left|\begin{array}{lllllllll} \text{最大化} & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & \\ \text{条　件} & a_1x_1 & + & a_2x_2 & + & \cdots & + & a_nx_n & \leq \quad b \\ & & & 0 \leq x_j \leq u_j, & & x_j \in \mathbb{Z}, & & j = 1,\, 2,\, \ldots,\, n. & \end{array}\right.$$

ただし，$\mathbb{Z}$ は整数全体の集合を表し，u_j は品物 G_j の個数を示す正の整数である．この問題を，動的計画法，あるいは分枝限定法によって解く方法を工夫しなさい．

6.4 土浦から上野まで最も割高に行く方法を提案しなさい．