アルゴリズムとデータ構造

第7回

アルゴリズムの設計
（§5.2分割統治法，§5.3動的計画法）

塩浦昭義

情報科学研究科 准教授

shioura@dais.is.tohoku.ac.jp
http://www.dais.is.tohoku.ac.jp/~shioura/teaching

中間試験について

- 日時：11月30日（水）13:00～14:30
- 今日までにレポートを一度も出していない場合，受験不可
- 教科書，ノート等の持ち込みは一切不可
- 座席はこちらで指定
- 試験内容：先週（第6回目）までの講義で教えたところ
  - アルゴリズムやデータ構造の挙動
  - 時間計算量の解析，および関連する証明問題
  - 用語の定義，など
- 50点満点，30点以上は合格
  - レポート提出状況が良い場合は，25点以上で合格
- 今後の予定：
  - 11月23日：祝日，11月30日：中間試験
  - 12月7日：休講，12月14日：授業

レポート（プログラム作成）

連結リストを用いた下記のプログラムを作成しなさい．

問1：キーボードから入力された数字を，次々に連結リストに追加していくプログラムを作成しなさい．ただし，入力される数字は正の整数とし，0が入力されたらプログラムは終了するものとする．

問2：（問1の続き）連結リストのセルの順番を，逆順に並び替えるプログラムを作成しなさい．ただし，並び替えの際には元々のセルを使うこととし，新しいセルは作らないこと．

※スライド「要素の追加の実装」に書いてあるプログラムを参考にしてください．

問3：（問1の続き）リストの中のセルを数字の小さい順に並び替えるプログラムを作成しなさい．出来れば，並び替えの際には元々のセルを使うこととし，新しいセルは作らないこと．

締切：11月30日（水）まで

復習＋先週の残り

ナップサック問題

- ハイキングの準備
- n個の品物の中から持って行くものを選択
- ナップサックには b kg まで入れられる
- 品物 $i = 1,2,\ldots,n$ の重さは a_i kg，利用価値は c_i

（共に正の実数）

- 利用価値の合計を最大にしたい

目的関数：$\sum_{i=1}^{n} c_i\, x_i$ → 最大

制約条件：$\sum_{i=1}^{n} a_i\, x_i \leq b$

$x_1, x_2, \ldots, x_n \in \{0,1\}$

0-1ナップサック問題

$x_j \in \{0,1\}$を$0 \leq x_j \leq 1$に置き換え

（例：品物が液体，粉末の場合）

連続ナップサック問題

連続ナップサック問題の最適解

連続ナップサック問題

目的関数: $\sum_{i=1}^{n} c_i\, x_i$ → 最大

制約条件: $\sum_{i=1}^{n} a_i\, x_i \leq b$

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

定理:

$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$ が成り立つと仮定

➔次のベクトル$(x_1^*, x_2^*, \ldots, x_n^*)$は最適解

$$x_j^* = \begin{cases} 1 & (j = 1,2,\ldots,q-1) \\ (b - \sum_{i=1}^{q-1} a_i)/a_q & (j = q) \\ 0 & (j = q+1, q+2, \ldots, n) \end{cases}$$

ただし, $q \in \{1,2,\ldots,n\}$は$\sum_{j=1}^{q-1} a_j \leq b < \sum_{j=1}^{q} a_j$ を満たす整数

連続ナップサック問題の最適解：例

定理：次のベクトル$(x_1^*, x_2^*, \ldots, x_n^*)$は最適解

$$x_j^* = \begin{cases} 1 & (j = 1,2, \ldots, q-1) \\ b - \sum_{i=1}^{q-1} a_i / a_q & (j = q) \\ 0 & (j = q+1, q+2, \ldots, n) \end{cases}$$

ただし，$q \in \{1,2, \ldots, n\}$は$\sum_{j=1}^{q-1} a_j \leq b < \sum_{j=1}^{q} a_j$ を満たす整数

	1	2	3	4	5	6	7	8
c_j	15	100	90	60	40	15	10	1
a_j	2	20	20	30	40	30	60	10

b＝102

a_jの合計72

q

よって(1,1,1,1,(102-72)/40,0,0,0)は最適解

最適解の計算（その1）

簡単な方法

ステップ1：c_j/a_j を大きい順にソートする --- O(n log n)時間

➔$\frac{c_{\pi(1)}}{a_{\pi(1)}} \geq \frac{c_{\pi(2)}}{a_{\pi(2)}} \geq \cdots \geq \frac{c_{\pi(n)}}{a_{\pi(n)}}$

$(\pi(1), \ldots, \pi(n)$は$1, \ldots, n$を並べ替えたもの$)$

ステップ2：$\sum_{j=1}^{q-1} a_{\pi(j)} \leq b < \sum_{j=1}^{q} a_{\pi(j)}$ を満たす

$q \in \{1,2, \ldots, n\}$を求める --- O(n)時間

ステップ3：定理に従って，最適解を計算 --- O(n)時間

合計で O(n log n)時間

最適解の計算（その2）

実はソートの必要はない！（以下，c_j/a_j の値は全て異なると仮定）

ステップ1：$\sum_{j\in S} a_j \le b < \sum_{j\in S} a_j + a_k$ $\left(S = \left\{j \middle| c_j/a_j > c_k/a_k\right\}\right)$
を満たす$k \in \{1,2,\ldots,n\}$を求める

--- O(n)時間で可能！

ステップ2：下記の式にしたがって最適解を計算 --- O(n)時間

$$x_j^* = \begin{cases} 1 & (j \in S\text{のとき}), \\ b - \sum_{i\in S} a_i \,/a_k & (j = k) \\ 0 & (c_j/a_j < c_k/a_k \text{ のとき}) \end{cases}$$

k の計算方法

アイディア：

$\frac{c_j}{a_j}$ の $h = \left\lceil \frac{n}{2} \right\rceil$ 番目に小さい要素を求める➔ $\frac{c_h}{a_h}$

$J_+ = \left\{ j \middle| \frac{c_j}{a_j} > \frac{c_h}{a_h} \right\}, J_- = \left\{ j \middle| \frac{c_j}{a_j} < \frac{c_h}{a_h} \right\}$ とおく

(1) $\sum_{j \in J_+} a_j > b$ ならば $k \in J_+$

(2) $\sum_{j \in J_+} a_j \leq b < \sum_{j \in J_+} a_j + a_h$ ならば $k = h$

(3) $b \geq \sum_{j \in J_+} a_j + a_h$ ならば $k \in J_-$

(1)の場合，J_+ の中で再帰的に k を探索（探索範囲は半分）

（3）の場合，J_- の中で再帰的に k を探索（探索範囲は半分）

➔合計でO(n)時間で k を計算できる

分割統治法

分割統治法 (divide-and-conquer method)

- アルゴリズム設計法のひとつ
- 以下の手順からなる
  1. 元の問題を小規模な部分問題に分割
  2. 部分問題を再帰的に解く
  3. 部分問題の解を統合することにより，元の問題の解を得る
- これまでに出てきた例：マージソート，クイックソート
- 新しい例：長大数のかけ算

分割

統治

マージソートと分割統治法

- マージソートは分割統治法に基づくアルゴリズム

①与えられた配列を2分割

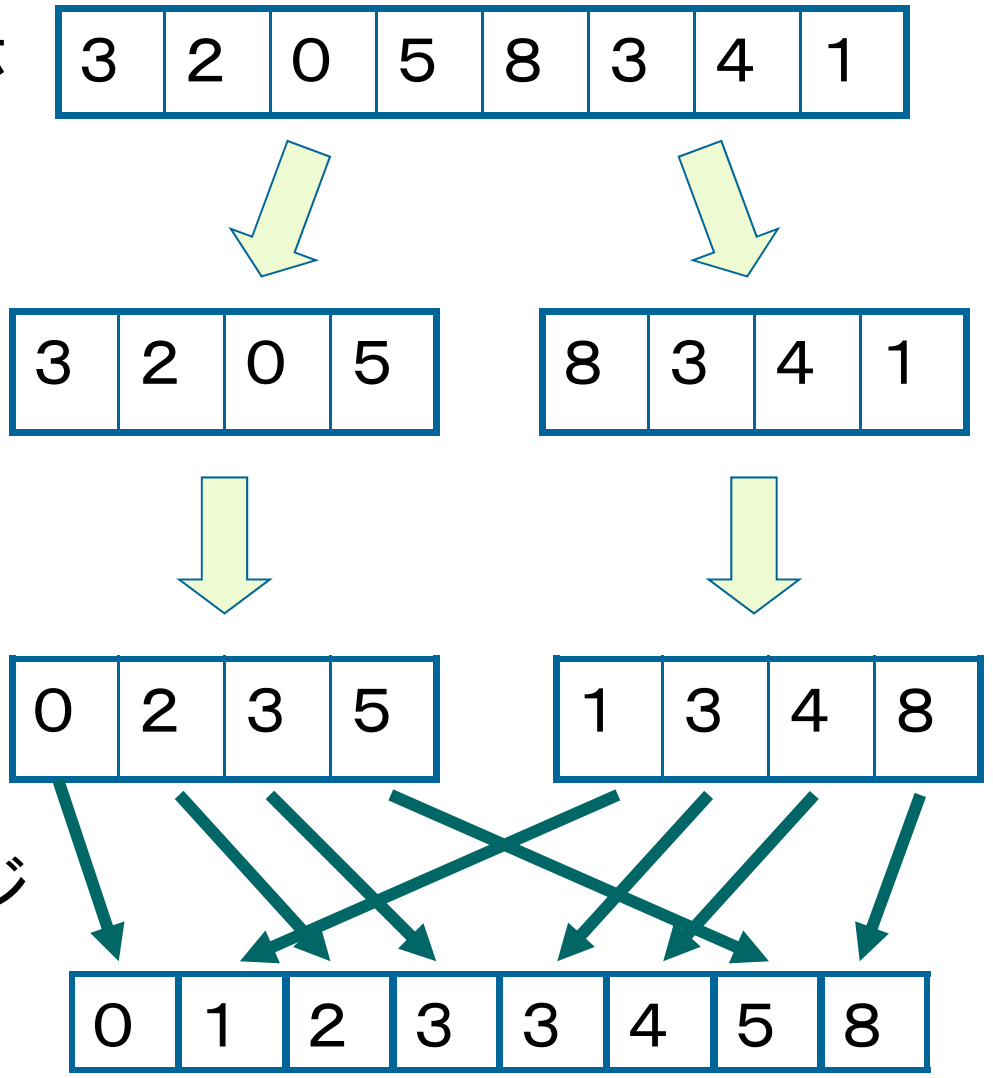
元の問題を
部分問題に分割

②2分割された配列を
それぞれ再帰的にソート

部分問題を再帰的に解く

③ソートされた2つの配列をマージ

部分問題の解を統合



クイックソートと分割統治法

- クイックソートは分割統治法に基づくアルゴリズム
  （と見ることも出来る）

①軸要素を選んで，軸要素未満の要素とそれ以外に分割

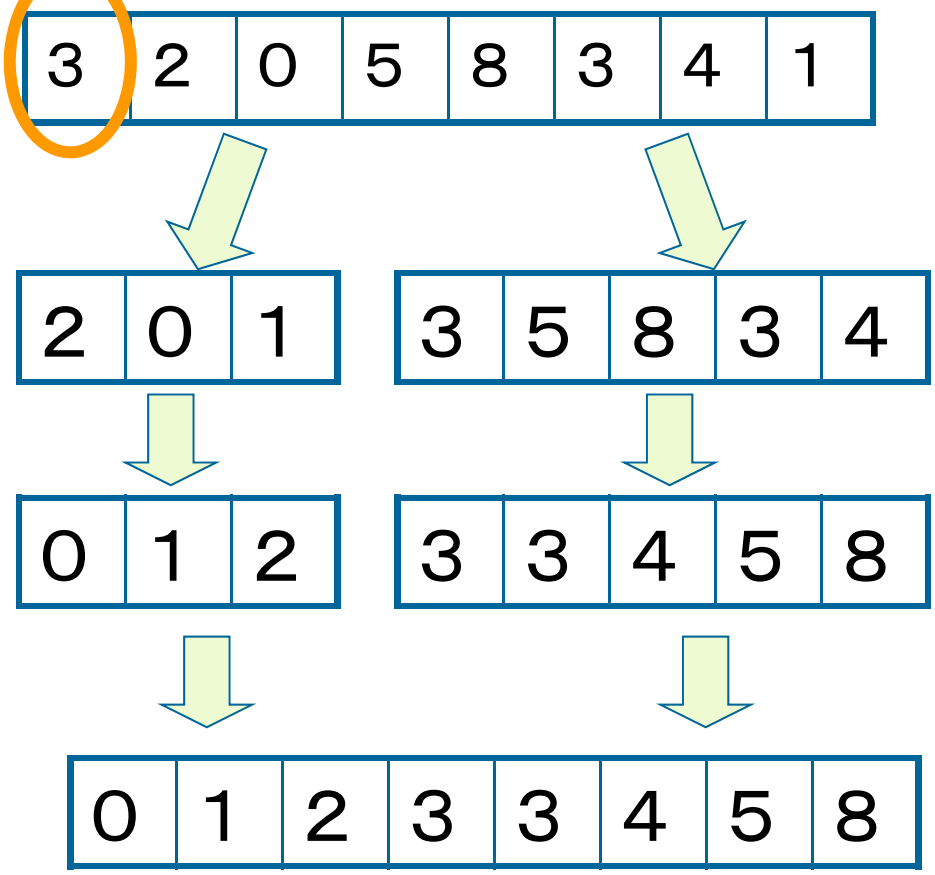
元の問題を部分問題に分割

②2分割された配列をそれぞれ再帰的にソート

部分問題を再帰的に解く

③ソートされた2つの配列をつなげる

部分問題の解を統合



長大数のかけ算

- 10進数のn桁の数 $x=x_1x_2\ldots x_n$ と $y=y_1y_2\ldots y_n$ のかけ算を，1桁同士のかけ算を使って計算
- 普通の計算方法：
  - 1桁同士のかけ算がn^2回必要
  - さらに繰り上げの計算，足し算，桁シフトが必要
  - 桁シフト：234 ➔ 234000 のように桁を移動

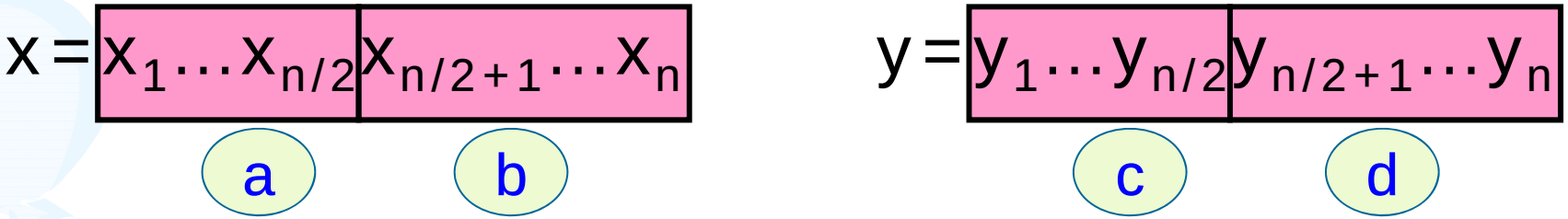
注意：かけ算は他の演算に比べて時間がかかる

- 分割統治に基づく方法：
  - 1桁同士のかけ算を $n^{\log_2 3} \simeq n^{1.59}$ に減らすことが可能

```
  583
×174
------
   12=3x4
  32 =8x4
20   =5x4
   21=3x7
      :
      :
```

長大数のかけ算に対する分割統治法

- 以下，説明を簡単にするために $n = 2^k$ と仮定
- x と y を n/2 桁ごとに2分割する



$$
\begin{aligned}
xy &= (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d) \\
&= ac \cdot 10^{n} + (ad + bc) \cdot 10^{n/2} + bd \\
&= ac \cdot 10^{n} + [(ac + bd) - (a - b)(c - d)] \cdot 10^{n/2} + bd
\end{aligned}
$$

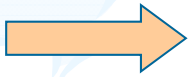
ac, bd, (a-b)(c-d) のかけ算の結果がわかれば，
あとはたし算と桁シフトのみを使って計算可能

n桁同士のかけ算➔n/2桁同士のかけ算3回　に分割

長大数のかけ算の時間計算量

- T(n)：n桁同士のかけ算の際に必要な，
  1桁同士のかけ算の回数
- 明らかに T(1) ＝ 1

n桁同士のかけ算➔n/2桁同士のかけ算3回　に分割



T(n) ＝ 3 T(n/2)

$$T(n) = 3 \cdot T(\tfrac{n}{2}) = 3^2 \cdot T(\tfrac{n}{2^2}) = 3^3 \cdot T(\tfrac{n}{2^3}) = \cdots$$

$$= 3^{\log_2 n} \cdot T(1) = n^{\log_2 3}$$

※nが2のべき乗でない場合も，同様のやり方で同様の結果が得られる

※一般に r 進数に対しても，同様の結果が成り立つ．

動的計画法

部分和問題 (subset-sum problem)

- 入力: n+1 個の正整数 $(a_1, a_2, \ldots, a_n;\ b)$
- 出力: yes または no を答える

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

を満たす0-1ベクトルが存在する➔yes, しない➔no

（別の見方: $a_1, a_2, \ldots, a_n$の幾つかを足し合わせてbにできる➔yes,
どう組合せても出来ない➔no）

例1: $a_1 = 5, a_2 = 3, a_3 = 2,\ b = 7$のとき,
$a_1 + a_3 = b$ が成り立つ ➔答えは yes

例2: $a_1 = 5, a_2 = 3, a_3 = 2,\ b = 6$ のとき,
a_1, a_2, a_3をどのように組み合わせても b に一致しない
➔答えは no

部分和問題の解の性質

部分和問題 $(a_1, a_2, \ldots, a_n;\ b)$ は

- $x_n = 1$ を満たす解をもつ

  ←→部分和問題 $(a_1, a_1, \ldots, a_{n-1};\ b - a_n)$ は解をもつ

- $x_n = 0$ を満たす解をもつ

  ←→部分和問題 $(a_1, a_2, \ldots, a_{n-1};\ b)$ は解をもつ

元の問題$(a_1, a_2, \ldots, a_n;\ b)$が解をもつ

2つの部分問題

$(a_1, a_1, \ldots, a_{n-1};\ b - a_n), (a_1, a_2, \ldots, a_{n-1};\ b)$

のどちらかは解をもつ

部分和問題の解法

解の性質を使うと，部分問題を繰り返し解くことで
元の問題の解を得ることが可能

部分問題$(a_1;\ p)$を$0 \leqq p \leqq b$について解く

部分問題$(a_1, a_2;\ p)$を$0 \leqq p \leqq b$について解く

部分問題$(a_1, \ldots, a_{n-1};\ p)$を$0 \leqq p \leqq b$について解く

部分問題$(a_1, \ldots, a_{n-1}, a_n;\ p)$を$0 \leqq p \leqq b$について解く

一つの部分問題は定数時間で解ける➔時間計算量O(nb)

部分和問題の解法の例

部分和問題(3,7,5,8,2; 11) を解く

部分問題(3; p) を 0≦p≦11 について解く

k＼p	0	1	2	3	4	5	6	7	8	9	10	11
1	○	×	×	○	×	×	×	×	×	×	×	×
2												
3												
4												
5												

p=0 のときは
常に
解をもつ

$p=a_1$ のときは
解をもつ
それ以外は
解をもたない

部分和問題の解法の例

部分和問題(3,7,5,8,2; 11) を解く

部分問題(3, 7; p) を 0≦ p ≦11 について解く

k＼p	0	1	2	3	4	5	6	7	8	9	10	11
1	○	×	×	○	×	×	×	×	×	×	×	×
2	○	×	×	○	×	×	×	○	×	×	○	×
3												
4												
5												

(a_1; p)が解をもつならば,
(a_1,a_2; p)も解をもつ

(a_1; p-a_2)が解をもつならば,
(a_1,a_2; p)も解をもつ

部分和問題の解法の例

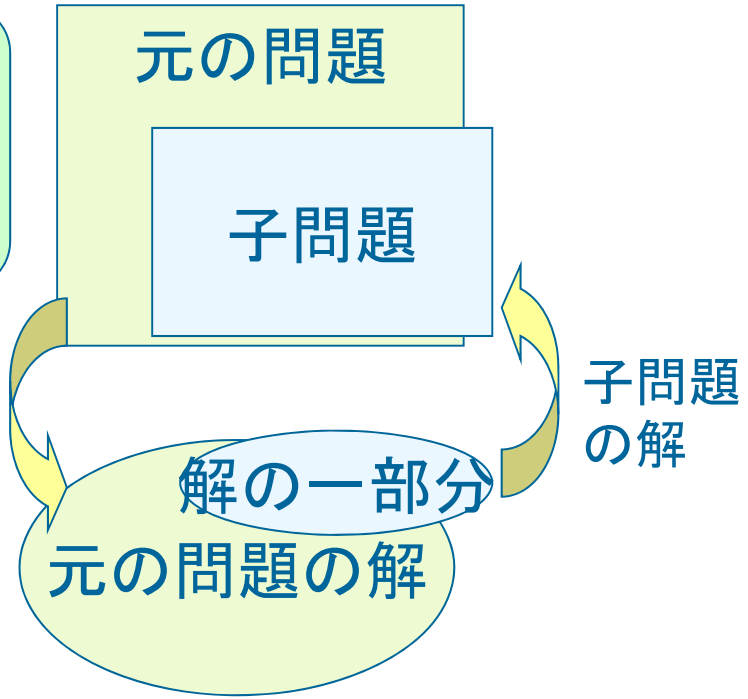
部分和問題(3,7,5,8,2; 11) を解く

k＼p	0	1	2	3	4	5	6	7	8	9	10	11
1	○	×	×	○	×	×	×	×	×	×	×	×
2	○	×	×	○	×	×	×	○	×	×	○	×
3	○	×	×	○	×	○	×	○	○	×	○	×
4	○	×	×	○	×	○	×	○	○	×	○	○
5	○	×	○	○	×	○	×	○	○	○	○	○

動的計画法

「最適性の原理」
元の問題の解の部分解は，
元の問題の部分問題の解である

再帰的なアルゴリズム



このアプローチ，もしくはこのアプローチにより得られる
再帰的アルゴリズムを動的計画法という

0-1ナップサック問題と部分問題

動的計画法は0-1ナップサック問題にも適用できる

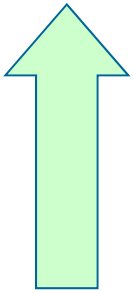
- 0-1ナップサック問題（入力は全て正の整数）

目的関数: $\sum_{i=1}^{n} c_i\, x_i$ → 最大
制約条件: $\sum_{i=1}^{n} a_i\, x_i \leq b$
$x_1, x_2, \ldots, x_n \in \{0,1\}$

元の問題の最適値
$f_n(b)$

- 部分問題（$k = 1, 2, \ldots, n,\ p = 0, 1, \ldots, b$）

$P_k(p)$: 目的関数: $\sum_{i=1}^{k} c_i\, x_i$ → 最大
制約条件: $\sum_{i=1}^{k} a_i\, x_i \leq p$
$x_1, x_2, \ldots, x_k \in \{0,1\}$



この部分問題の最適値
$f_k(p)$

0-1ナップサック問題の最適解の性質

- 部分問題$(k = 1, 2, \ldots, n,\ p = 0, 1, \ldots, b)$

$P_k(p)$：目的関数：$\sum_{i=1}^{k} c_i\, x_i$ → 最大
制約条件：$\sum_{i=1}^{k} a_i\, x_i \leq p$
$x_1, x_2, \ldots, x_k \in \{0,1\}$

この部分問題の最適値

$f_k(p)$

- $P_k(p)$の最適解$(x_1^*, \ldots, x_{k-1}^*, x_k^*)$が $x_k^* = 0$ を満たす
  → $(x_1^*, \ldots, x_{k-1}^*)$は$P_{k-1}(p)$の最適解，　$f_k(p) = f_{k-1}(p)$
- $P_k(p)$の最適解$(x_1^*, \ldots, x_{k-1}^*, x_k^*)$が $x_k^* = 1$ を満たす
  → $(x_1^*, \ldots, x_{k-1}^*)$は$P_{k-1}(p - a_k)$の最適解，

$$f_k(p) = f_{k-1}(p - a_k) + c_k$$

∴ $P_k(p)$の最適解は，$P_{k-1}(p)$の最適解
または$P_{k-1}(p - a_k)$の最適解から作ることが出来る

0-1ナップサック問題に関する再帰式

- $x_k^* = 0$ ➔ $f_k(p) = f_{k-1}(p)$
- $x_k^* = 1$ ➔ $f_k(p) = f_{k-1}(p - a_k) + c_k$

$$f_k(p) = \max\{f_{k-1}(p),\ f_{k-1}(p - a_k) + c_k\}$$

$$\text{ただし } f_1(p) = \begin{cases} 0 & (0 \le p \le a_1) \\ c_1 & (a_1 \le p \le b) \end{cases}$$

動的計画法

上記の再帰式をつかって，

$f_2(0), f_2(1), \ldots, f_2(b),\ f_3(0), f_3(1), \ldots, f_3(b),$

$\ldots,\ f_n(0), f_n(1), \ldots, f_n(b)$

を順に計算 $\Longrightarrow$ 元問題の最適値が得られる

計算時間

O(nb)

0-1ナップサック問題の最適解の計算

- 最適解も再帰的に計算可能

$$f_k(p) = \max\{f_{k-1}(p),\ f_{k-1}(p - a_k) + c_k\}$$

このとき，

- $f_k(p) = f_{k-1}(p)$ ならば
  $P_k(p)$の最適解は$(x_1^*, \ldots, x_{k-1}^*, 0)$
  （ただし，$(x_1^*, \ldots, x_{k-1}^*)$は$P_{k-1}(p)$の最適解）

- $f_k(p) = f_{k-1}(p - a_k) + c_k$ ならば
  $P_k(p)$の最適解は$(x_1^*, \ldots, x_{k-1}^*, 1)$
  （ただし，$(x_1^*, \ldots, x_{k-1}^*)$は$P_{k-1}(p - a_k)$の最適解）

レポート問題（締切：12月14日）

問1：この0-1ナップサック問題を動的計画法で解け．下記のような表を作成すること．

最大化 $10x_1 + 7x_2 + 25x_3 + 24x_4$

条件 $2x_1 + x_2 + 6x_3 + 5x_4 \leq 7$

$x_j \in \{0, 1\}\ (j = 1, 2, 3, 4)$

k＼p	0	1	2	3	4	5	6	7
1	0	0	10	10	10	10	10	10
2								
3								
4								

問2：x＝4782 と y＝9175 のかけ算を，分割統治法を使って計算せよ．再帰的な計算の部分も省略せずに書くこと．