

アルゴリズムと データ構造

第6回

§ 6.1 簡単な最適化問題

塩浦昭義

情報科学研究科 准教授

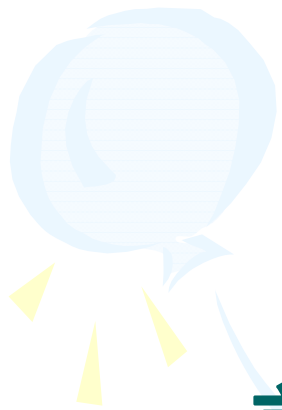
shioura@dais.is.tohoku.ac.jp

<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 (予定)
- 11/16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: 今日(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, 29点以下は追試レポートもしくは単位不可
- 今後の予定
 - 11/16: 第7回, 11/23: 祝日



前回の復習 第P要素の選択のアルゴリズム

第p要素の選択

- n 個の実数の中から p 番目に小さい(大きい)ものを選ぶ
 - $p=1 \rightarrow$ n個の実数の中の**最小値**
 - $p=n \rightarrow$ n個の実数の中の**最大値**
 - $p = \lceil n/2 \rceil$ または $\lfloor n/2 \rfloor \rightarrow$ n個の実数の中の**中央値(median)**

最小値

中央値
(5番目に小さい値)

73 25 69 52 91 37 76 40 88

最大値

2番目に
大きい値

SELECTの手順(その1)

p=5 の場合

(0) $A[1], \dots, A[n]$ が全て同じ要素
いずれかの要素を出力, 終了

3	3	3	3	3	3	3	3
---	---	---	---	---	---	---	---

(1) $A[1], \dots, A[n]$ から
ひとつの値(軸要素)を選ぶ

3	2	0	5	8	3	4	1
---	---	---	---	---	---	---	---

(2) 軸要素未満の要素と
それ以外に分割
 $k =$ 軸要素未満の要素数

2	0	1
---	---	---

$k=3$

3	5	8	3	4
---	---	---	---	---

$n-k=5$

SELECTの手順(その2)

(3) $p \leq k$ のとき,

軸要素未満の配列から第 p 要素を再帰的に選択

$p > k$ のとき,

軸要素以上の配列から第 $p-k$ 要素を再帰的に選択

$p=5 > 3=k$ なので
右の配列から
第 $5-3=2$ 要素を
再帰的に選択

$p=5$ の場合

2	0	1
---	---	---

$k=3$

3	5	8	3	4
---	---	---	---	---

$n-k=5$

3

SELECTでの軸要素の選び方

- 説明を簡単にするため、全ての要素が異なると仮定
 - $A[i]$ と $A[k]$ が等しい場合は、
 - 配列の番号 i, k によって大小関係を決めればよい
($i < k$ ならば $A[i]$ が「小さい」、 $i > k$ ならば $A[k]$ が「小さい」)

手順1: n 個の要素を5個ずつのグループに分ける。

5個未満のグループが一つ出来たら、それは別にしておく。

→ ちょうど5個のグループの数は $\lfloor n/5 \rfloor$

1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

$$n = 29$$

$$\lfloor n/5 \rfloor = 5$$

SELECTでの軸要素の選び方

手順2: $\lfloor n/5 \rfloor$ 個の各グループにおいて、中央値(3番目に小さい値)を計算し、取り出す。

手順3: 手順2で取り出した $\lfloor n/5 \rfloor$ 個の要素に対し、中央値($\lceil \lfloor n/5 \rfloor / 2 \rceil = \lfloor (n+5)/10 \rfloor$ 番目に小さい値)を計算し、軸要素 a とする

1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

$$n = 29$$

$$\lfloor n/5 \rfloor = 5$$

中央値の集合

13, 49, 70, 28, 20

この中の中央値は

28 ← 軸要素 a

各反復での要素数の減少率

グループ1, 2のそれぞれの中央値は軸要素 a 未満
 →グループ1, 2には a未満の要素が3つ以上存在
 →a 未満の要素数
 $\geq 3 \times 3 \text{グループ} - 1 = 8$

グループ4, 5のそれぞれの中央値は軸要素aより大きい
 →グループ4, 5には aより大きい要素が3つ以上存在
 →a より大きい要素の数
 $\geq 3 \times 3 \text{グループ} - 1 = 8$

グループ1	グループ2	グループ3	グループ4	グループ5	
1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

軸要素 a より大きい要素の割合は？
 軸要素 a より小さい要素の割合は？

各反復での要素数の減少率

軸要素 a は, 各グループの中央値の中の中央値, $\lceil \lfloor n/5 \rfloor / 2 \rceil = \lfloor (n+5)/10 \rfloor$ 番目に小さい要素

「グループの中央値 $< a$ 」となるグループ数は $\lfloor (n+5)/10 \rfloor - 1$
 →各グループには a 未満の要素が3つ以上存在

→ a 未満の要素数

$$\geq 3 \times \lfloor (n+5)/10 \rfloor - 1$$

$n \geq 50$ のとき, この値は $n/4$ 以上

$n \geq 50$, かつ見つけたい要素が軸要素以上 →

次の反復での要素数 $\leq 3/4n$

グループ1 グループ2 グループ3 グループ4 グループ5

1	8	7	40	5	23
3	16	22	43	37	33
13	20	28	49	70	50
76	25	30	52	75	77
99	37	47	57	85	

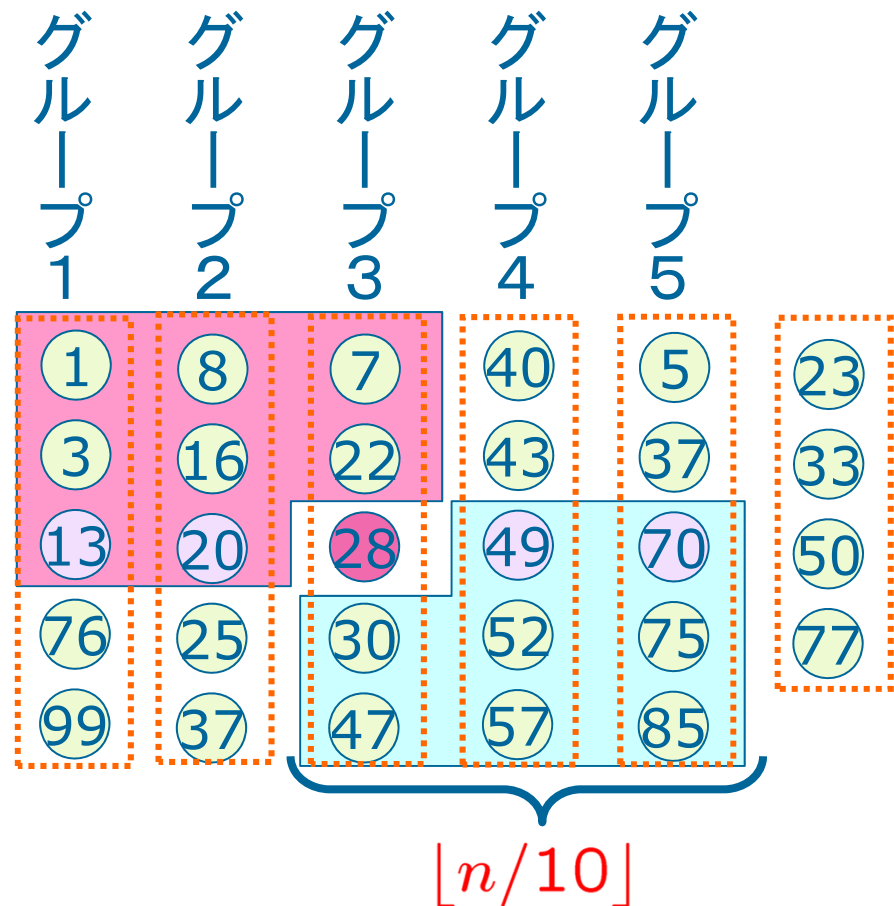
$$\lfloor (n+5)/10 \rfloor$$

各反復での時間計算量の解析

軸要素 a は, 各グループの中央値の中の中央値, $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ 番目に大きい要素

「グループの中央値 $> a$ 」となるグループ数は $\lfloor n/10 \rfloor - 1$
 →各グループには a より大きい要素が3つ以上存在
 → a より大きい要素の数
 $\geq 3 \times \lfloor n/10 \rfloor - 1$
 $n \geq 80$ のとき, この値は $n/4$ 以上

$n \geq 80$, かつ見つけたい要素が軸要素未満 →
 次の反復での要素数 $\leq 3/4n$

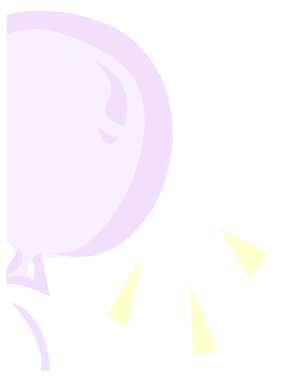


SELECTの時間計算量の解析

- アルゴリズムSELECTでは,
 - $n \geq 80$ のとき, 一回の反復で要素数が n から $3/4n$ 以下に減少する
 - $n < 80$ のとき, 定数時間で第 p 要素を選択することが可能 (要素数 n が定数以下なので)
- 反復回数を k とおく
 - $(3/4)^{k-1} \geq 80, (3/4)^k < 80$ を満たす
 - 合計の時間計算量は
$$c\{n + 3/4n + (3/4)^2n + \dots + (3/4)^{k-2}n + 80\}$$
$$\leq c \times 4n + 80 \quad c = O(n)$$



今日の内容

- 2つの最適化問題に対するアルゴリズム
 - ▶ 分離凸資源配分問題
 - ▶ 連続ナップサック問題
 - これまでに学んだ基本的なアルゴリズム・データ構造を利用して最適化問題を解くことができる
 - アプローチ:
 - ▶ それぞれの問題の最適解の満たすべき条件を明確にする(数学)
 - ▶ 条件を満たす解をアルゴリズムにより求める(情報科学)
- 

資源配分問題

- 限られた資源をいくつかのグループ・人・組織に配分
- 経費・損失を最小化, または利益・満足度を最大化
- 離散的な資源 (車, PC, 人など) の配分問題の定式化

目的関数 $f(x_1, x_2, \dots, x_n) \rightarrow$ 最小 (または最大)

制約条件 $x_1 + x_2 + \dots + x_n = N$

x_j : 非負整数, $j = 1, 2, \dots, n$

- 最適化問題の用語
 - 実行可能解 (許容解): 全ての条件を満たす解
 - 最適解: 実行可能解の中で目的関数を最小化 (または最大化) するもの

目的関数が分離凸な 資源配分問題

- 1変数関数 $f: \mathbb{Z} \rightarrow \mathbb{R}$ は **凸関数** (下に凸)
↔ 差分関数 $d(\alpha) = f(\alpha) - f(\alpha - 1)$ が α に関して **単調非減少**
 $\dots \leq d(-2) \leq d(-1) \leq d(0) \leq d(1) \leq d(2) \leq \dots$

- 例: $f(\alpha) = x^2$ は凸関数

- n 変数関数 $f: \mathbb{Z}^n \rightarrow \mathbb{R}$ は **分離凸関数**

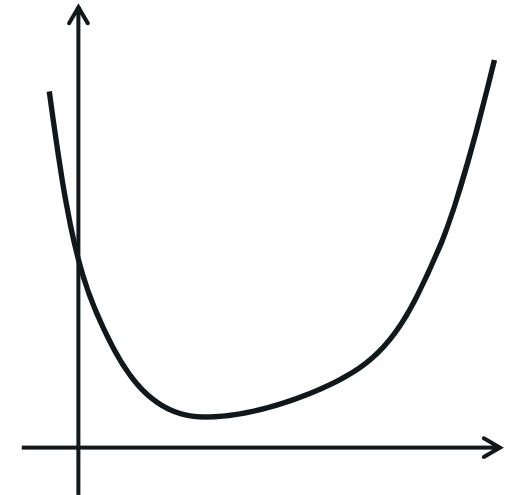
↔ $f(x_1, x_2, \dots, x_n)$ は
 $f_1(x_1) + f_2(x_2) + \dots + f_n(x_n)$
(各 f_j は凸関数) の形に書ける

- 目的関数が **分離凸関数** の場合の
資源配分問題

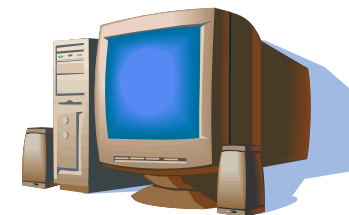
目的関数 $\sum_{j=1}^n f_j(x_j) \rightarrow$ 最小

制約条件 $x_1 + x_2 + \dots + x_n = N$

x_j : **非負整数**, $j = 1, 2, \dots, n$

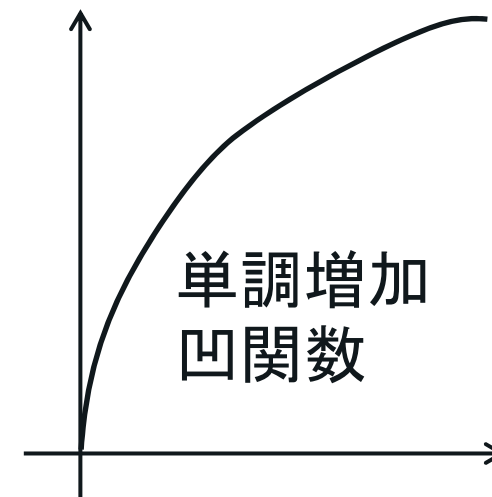


目的関数が分離凸な 資源配分問題の例1



- 情報知能システム総合学科でPC(またはiPad, デジカメ, etc.)を合計 N 台購入
- これを n か所の研究室に配分
- 研究室 j への配分数(変数): x_j , その際の満足度 $f_j(x_j)$
- f_j は単調増加, 凹関数(上に凸)
 - PCをもっていない人がPCを新たにもらったときの嬉しさ \geq PCを5台もっている人がPCを新たにもらったときの嬉しさ
- 満足度の合計値 $\sum_{j=1}^n f_j(x_j)$ を最大にする配分
= $\sum_{j=1}^n \{-f_j(x_j)\}$ を最小にする配分

これは分離凸関数



目的関数が分離凸な 資源配分問題の例2

- M県の国会議員定数 N を各選挙区 $j = 1, 2, \dots, n$ に配分
- 選挙区 j の人口 $p_j \rightarrow$ M県の総人口は $p = \sum_{j=1}^n p_j$
 - M県の人口一人当りの議員数は p/k
- 選挙区 j のへの定数配分(変数): x_j
 - 公平性のため, 選挙区 j での人口一人当りの議員数 p_j/x_j を出来るだけ p/k に近づけたい
 - \rightarrow 誤差の2乗の和を最小化

目的関数 $\sum_{j=1}^n p_j \left(\frac{x_j}{p_j} - \frac{p}{k} \right)^2 \rightarrow \text{最小}$

制約条件 $x_1 + x_2 + \dots + x_n = N$

x_j : 非負整数, $j = 1, 2, \dots, n$



分離凸資源配分問題： 最適解の必要十分条件

定理：分離凸資源配分問題の

実行可能解 (x_1, x_2, \dots, x_n) は最適解

↔ ある実数 λ が存在して、以下の条件を満たす

(i) $x_j > 0$ ならば $d_j(x_j) \leq \lambda$ (ii) $d_j(x_j + 1) \geq \lambda$

$$d_j(\alpha) = f_j(\alpha) - f_j(\alpha - 1)$$

証明の概略

[←の証明] 関数が凸であることと条件(i), (ii)を使うと、

「任意の実行可能解の目的関数値

\geq (i), (ii) を満たす実行可能解の目的関数値」を示せる。

[→の証明] 対偶を示す。

(i), (ii) を満たさない実行可能解は目的関数値を減らすことが可能

∴ 最適解ではない

最適解の必要十分条件: 例

$$n = 5, N = 6$$

$$f_1(\alpha) = \frac{1}{\alpha+1}, f_2(\alpha) = (\alpha - 2.5)^2, f_3(\alpha) = 1 + 0.1\alpha^3,$$

$$f_4(\alpha) = -1 + (\alpha - 2)^2, f_5(\alpha) = 3 - \sqrt{\alpha}$$

$(x_1, x_2, x_3, x_4, x_5) = (1, 2, 0, 2, 1)$ は最適解

$$d_1(1) = -0.5, d_2(2) = -2, d_4(2) = -1, d_5(1) = -1$$

$$d_1(2) = -0.17, d_2(3) = +2, d_3(1) = +0.1, d_4(3) = +1, d_5(2) = -0.41$$

→ $-0.5 \leq \lambda \leq -0.41$ を満たす λ は条件 (i), (ii) を満たす

$(x_1, x_2, x_3, x_4, x_5) = (3, 0, 1, 0, 2)$ は最適解ではない

$$d_1(3) = -0.08, d_3(1) = +0.1, d_5(2) = -0.41$$

$$d_1(4) = -0.05, d_2(1) = -4, d_3(2) = +0.7, d_4(1) = -3, d_5(3) = -0.32$$

→ 条件 (i), (ii) を満たす λ は存在しない

x_3 を1減らし, x_2 を1増やす

→ $(3, 1, 0, 0, 2)$ の方が目的関数値が小さい

アルゴリズムその1: 欲張り法

(0, 0, ..., 0)からスタート,

目的関数値をなるべく減らす(増やさない)ように,

いずれかの変数を1増やすことを繰り返す

ステップ0: $(x_1, \dots, x_n) = (0, \dots, 0)$ とおく.

ステップ1: $x_1 + \dots + x_n = N$ ならば終了. (x_1, \dots, x_n) は最適解

ステップ2: $d_j(x_j + 1)$ を最小にする j を選び, x_j を1増やす.

ステップ3: ステップ1に戻る.

欲張り法の例

$$n = 5, N = 6$$

$$f_1(\alpha) = \frac{1}{\alpha+1}, f_2(\alpha) = (\alpha - 2.5)^2, f_3(\alpha) = 1 + 0.1\alpha^3,$$

$$f_4(\alpha) = -1 + (\alpha - 2)^2, f_5(\alpha) = 3 - \sqrt{\alpha}$$

$d_j(x_j)$ の値	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
$x_j = 1$	-0.5	-4	+0.1	-3	-1
$x_j = 2$	-0.17	-2	+0.7	-1	-0.41
$x_j = 3$	-0.08	+2	+1.9	+1	-0.32

$(0,0,0,0,0) \rightarrow (0,1,0,0,0) \rightarrow (0,1,0,1,0) \rightarrow (0,2,0,1,0)$
 $\rightarrow (0,2,0,2,0) \rightarrow (0,2,0,2,1) \rightarrow (1,2,0,2,1)$

欲張り法の正当性

定理: 欲張り法の出力 (x_1^*, \dots, x_n^*) は最適解である.

証明: アルゴリズムが最後に増やした変数を x_k とする.
 $\lambda = d_k(x_k^*)$ とおく.

- f_k は凸関数 $\rightarrow \lambda \leq d_k(x_k^* + 1)$
 - ステップ2では, $d_j(x_j + 1)$ を最小にする j を選ぶ
 $\rightarrow x_j^* > 0$ ならば $\lambda \geq d_j(x_j^*)$
($\because x_k$ を x_k^* に増やす以前に, 変数 x_j は x_j^* になっている)
 - k 以外の $j = 1, 2, \dots, n$ に対して $\lambda \leq d_j(x_j^* + 1)$
($\because x_j$ を $x_j^* + 1$ に増やす以前に, 変数 x_k は x_k^* になっている)
- \therefore 最適解の必要十分条件より, (x_1^*, \dots, x_n^*) は最適解

欲張り法の計算時間

- 各反復では $\sum_{j=1}^n x_j$ の値が1ずつ増える
 $\sum_{j=1}^n x_j$ の初期値は0, これが N になった終了 → 反復回数 = N
- 各反復では, $d_j(x_j + 1)$ を最小にする j を選ぶ
 - 簡単なやり方では $O(n)$ 時間が必要
 - ヒープを使うと1回当り $O(\log n)$ 時間で可能

ヒープの使い方

- ヒープには常に n 個の値 $d_j(x_j + 1)$ ($j = 1, \dots, n$) を入れておく
- 最小値が $d_k(x_k + 1)$ の場合
 → これを削除, 代わりに $d_k(x_k + 2)$ を挿入
 一回当り $O(\log n)$ 時間で実行可能

定理: 欲張り法は $O(N \log n)$ 時間で動く.

擬多項式時間

→ さらに改良して $O(n + n \log N/n)$ 時間が可能

多項式時間

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 $i = 1, 2, \dots, n$ 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: $\sum_{i=1}^n c_i x_i \rightarrow \text{最大}$
制約条件: $\sum_{i=1}^n a_i x_i \leq b$
 $x_1, x_2, \dots, x_n \in \{0, 1\}$

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: $\sum_{i=1}^n c_i x_i \rightarrow \text{最大}$
制約条件: $\sum_{i=1}^n a_i x_i \leq b$
 $0 \leq x_j \leq 1 (j = 1, 2, \dots, n)$

定理:

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

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

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

ただし, $q \in \{1, 2, \dots, n\}$ 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 $(x_1^*, x_2^*, \dots, x_n^*)$ は最適解

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

ただし、 $q \in \{1, 2, \dots, n\}$ 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

連続ナップサック問題の最適解

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

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

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

証明の概略:

- $\sum_{i=1}^n a_i x_i = b$ を満たす最適解が必ず存在
- $\sum_{i=1}^n a_i x_i = b$ を満たす実行可能解について
 - ある j ($1 \leq j \leq q - 1$) に対して $x_j < 1$
→ ある x_i ($q \leq i \leq n$) を減らして x_j を増やすと関数値が増える
 - ある j ($q + 1 \leq j \leq n$) に対して $x_j > 0$
→ ある x_i ($1 \leq i \leq q$) を増やして x_j を減らすと関数値が増える
- 上記のベクトル $(x_1^*, x_2^*, \dots, x_n^*)$ 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

$$\rightarrow \frac{c_{\pi(1)}}{a_{\pi(1)}} \geq \frac{c_{\pi(2)}}{a_{\pi(2)}} \geq \dots \geq \frac{c_{\pi(n)}}{a_{\pi(n)}}$$

($\pi(1), \dots, \pi(n)$ は $1, \dots, n$ 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

$q \in \{1, 2, \dots, n\}$ 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: $\sum_{j \in S} a_j \leq b < \sum_{j \in S} a_j + a_k$ ($S = \{j \mid c_j/a_j > c_k/a_k\}$)

を満たす $k \in \{1, 2, \dots, n\}$ 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

アイデア:

$\frac{c_j}{a_j}$ の $h = \lfloor \frac{n}{2} \rfloor$ 番目に小さい要素を求める $\rightarrow \frac{c_h}{a_h}$

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

(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)

\rightarrow 合計で $O(n)$ 時間で k を計算できる

レポート問題

問1: 次の資源配分問題を欲張り法を使って解け. 各反復で増加させる変数の選び方について詳しく書くこと.

目的関数 $2x_1^2 + (x_2^3 - 2x_2) + e^{x_3} \rightarrow$ 最小

制約条件 $x_1 + x_2 + x_3 = 6, x_1, x_2, x_3$ は非負整数

問2: 次の連続ナップサック問題の最適解を求めよ. 計算の過程についても詳しく書くこと.

(1) 目的関数: $5x_1 + 50x_2 + 46x_3 + 64x_4 + 50x_5 + 50x_6 \rightarrow$ 最大
制約条件: $17x_1 + 75x_2 + 64x_3 + 80x_4 + 59x_5 + 56x_6 \leq 190$
 $0 \leq x_j \leq 1 (j = 1, 2, \dots, 6)$

(2)

目的関数: $39x_1 + 37x_2 + 7x_3 + 5x_4 + 10x_5 + 20x_6 + 70x_7 \rightarrow$ 最大
制約条件: $20x_1 + 19x_2 + 4x_3 + 3x_4 + 6x_5 + 10x_6 + 31x_7 \leq 50$
 $0 \leq x_j \leq 1 (j = 1, 2, \dots, 7)$