数理・計算科学特論 C 第 2 回

河瀬 康志

2021 年 6 月 15 日

アウトライン

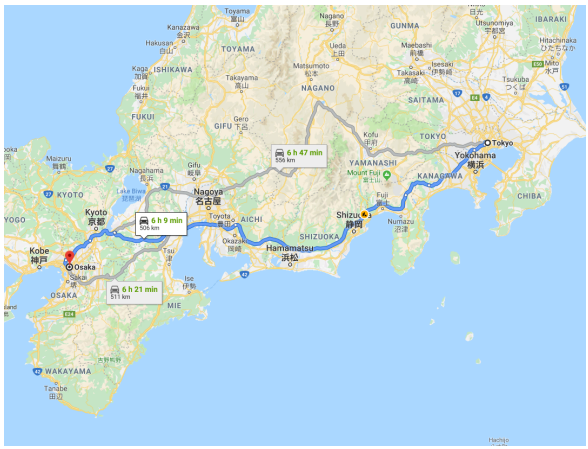
最適化とは

最適解

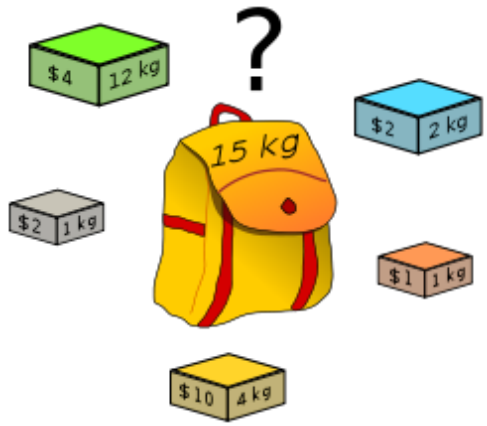
条件を満たす中で，ある指標で最も良いものを見つけること

- 最適解を見つけるための手続き (アルゴリズム) について考察
- 計算効率（計算時間，メモリ使用量）がよく，実装が容易なものを見つけられると嬉しい

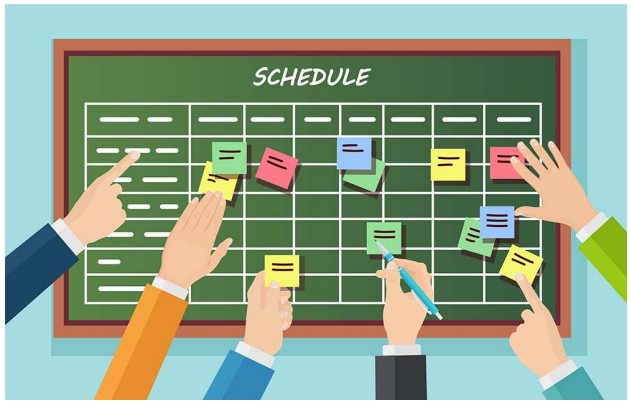
最適化問題の例



最短路問題

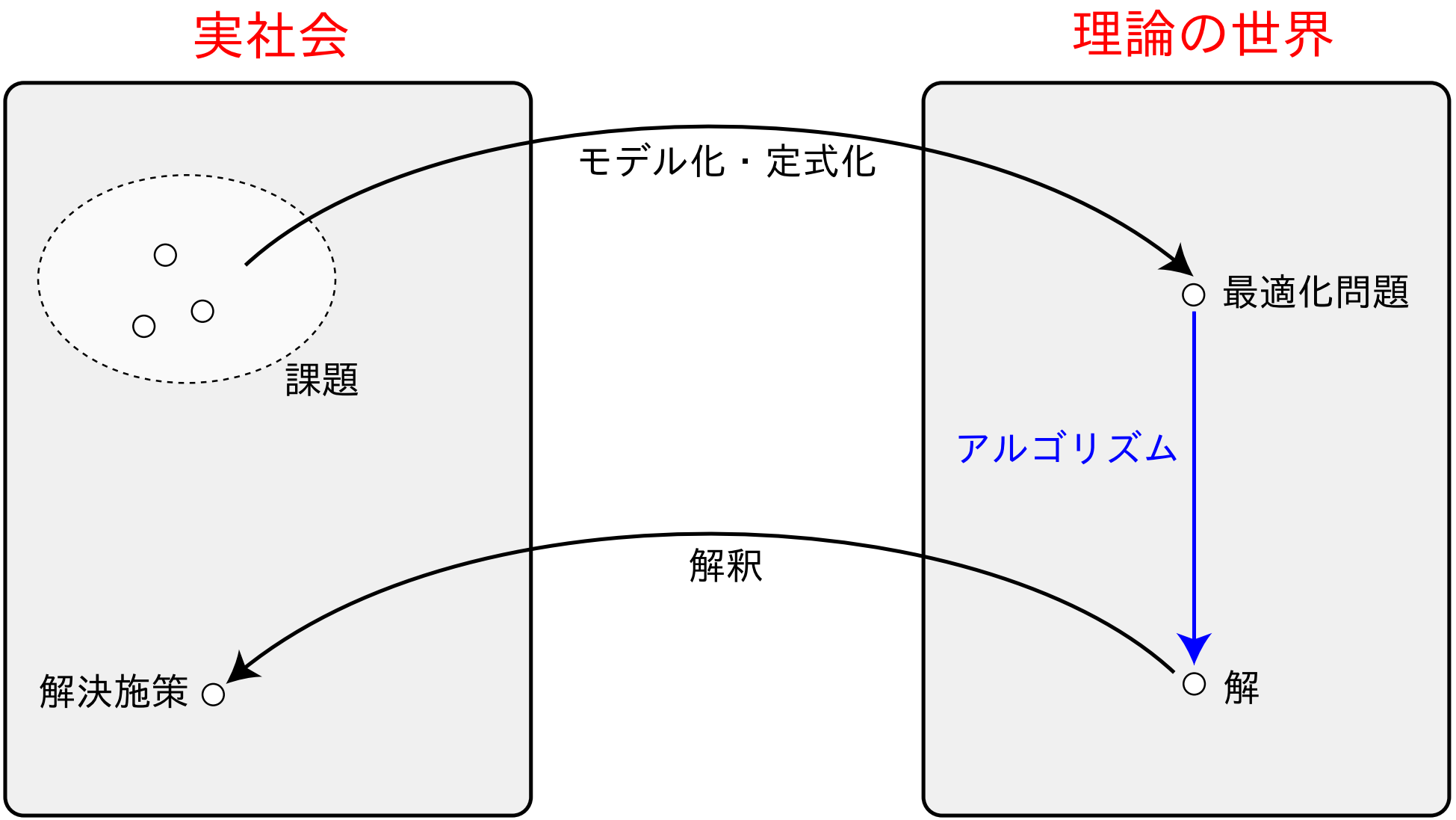


ナップサック問題



スケジューリング

最適化を用いた意思決定



オフライン vs オンライン

オフライン最適化

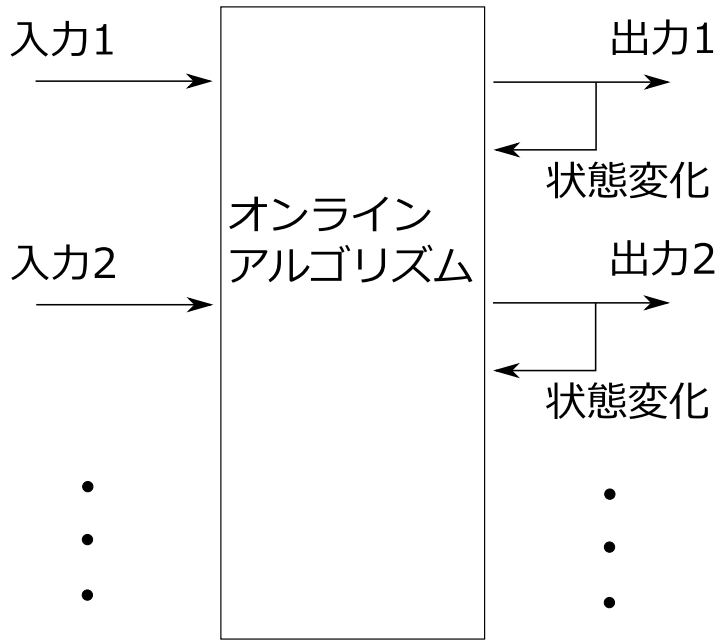
入力 一度に与えられる

目的 最適解を効率的に求める

オンライン最適化

入力 逐次的に与えられる

目的 なるべく良い選択



アウトライン

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

アルゴリズム 1: ずっとレンタルし続ける

- 100 回行くと 100 万円かかるが，最初に購入しておけば 5 万円
  → 比は $100/5 = 20$
- k 回行くとすると，比は $k/5 \to \infty$ $(k \to \infty)$

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

アルゴリズム 1: ずっとレンタルし続ける

- 100 回行くと 100 万円かかるが，最初に購入しておけば 5 万円
  → 比は $100/5 = 20$
- k 回行くとすると，比は $k/5 \to \infty$ $(k \to \infty)$

アルゴリズム 2: 最初に購入する

- 1 回しか行かない場合が最悪で，比は $5/1 = 5$

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

アルゴリズム 3: 4 回目まではレンタルで 5 回目に購入する

スキーに k 回行く場合：

- $k \leq 4$ のとき
  毎回レンタルが最適であり，比は 1
- $k \geq 5$ のとき
  最初に購入が最適であり，比は $(4+5)/5 = 1.8$

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

もっと良いアルゴリズムはあるか？ → ない

スキーレンタル問題

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと 5 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

もっと良いアルゴリズムはあるか？ → ない

- 一度購入したらレンタルをすることはないので，
「何回目に買うか」でアルゴリズムは決まる
- ℓ 回目に買うアルゴリズムを考える
- 最悪なのは ℓ 回しかスキーに行かない場合であり，比は

$$\frac{(\ell - 1) + 5}{\min\{\ell, 5\}} \geq \frac{9}{5} = 1.8$$

スキーレンタル問題 (一般の形)

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと N 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

- 最適なオンラインアルゴリズムは $N-1$ 回のレンタル後に購入
- 最悪の状況は N 回スキーに行く場合で，競合比は $\frac{(N-1)+N}{N} = 2 - \frac{1}{N}$

購入金額を超えない範囲でレンタルするのが最適

スキーレンタル問題 (一般の形)

これからスキーを始めるが，今後何回スキーにいくかわからない．
スキー板を買うべきか，レンタルすべきか？

- レンタルすると，1 回につき 1 万円
- 買うと N 万円
- 買ったスキーは，ずっと使い続けることができる（壊れない）

- 最適なオンラインアルゴリズムは $N-1$ 回のレンタル後に購入
- 最悪の状況は N 回スキーに行く場合で，競合比は $\frac{(N-1)+N}{N} = 2 - \frac{1}{N}$

購入金額を超えない範囲でレンタルするのが最適

アウトライン

競合比

スキーレンタル問題で考えた比は競合比と呼ばれており，
最小化問題の場合，次のように定義される

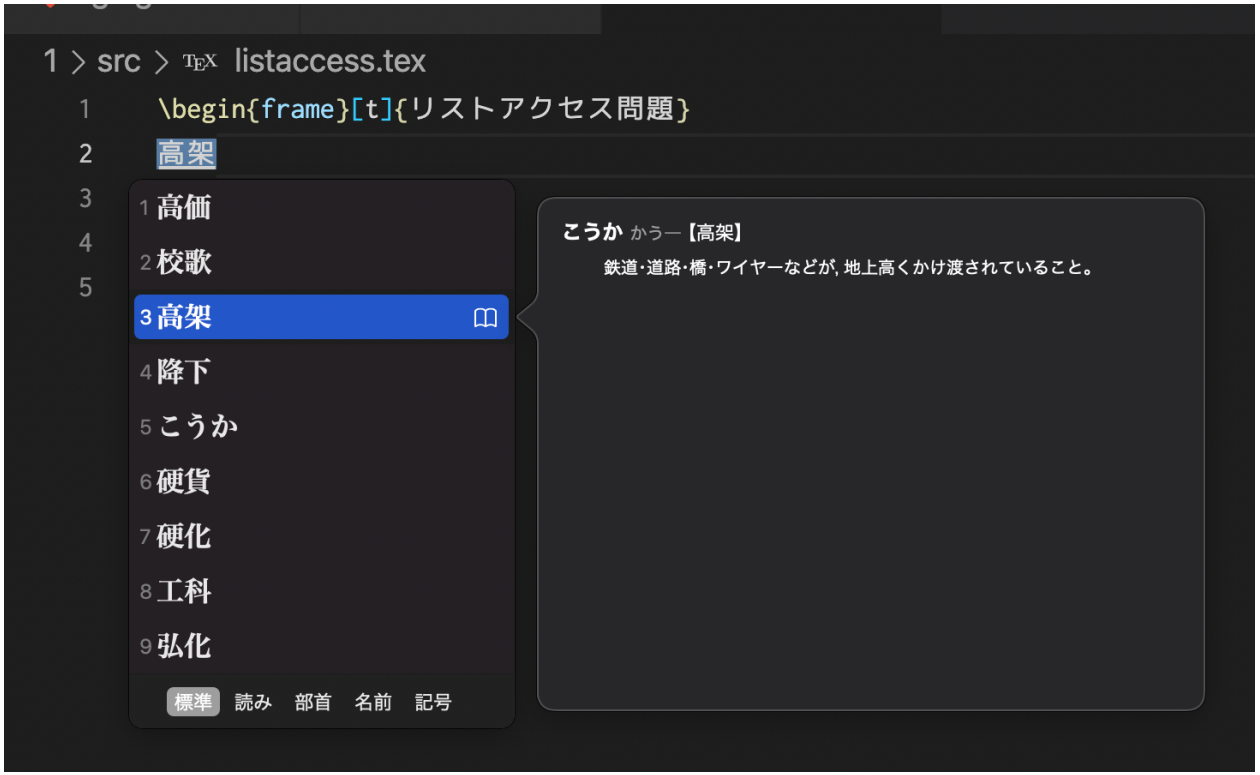
$$\sup_{\text{入力列}} \frac{\text{オンラインアルゴリズムでのコスト}}{\text{入力列を知っていた場合の最適コスト}}$$

- 最悪の場合に対し，最適コストの何倍以下に抑えられるかを保証
- 1 以上であり小さいほど嬉しい
- 競合比がなるべく小さいアルゴリズムを設計したい

最大化問題の場合は $\inf_{\text{入力列}} \frac{\text{オンラインアルゴリズムでの利得}}{\text{入力列を知っていた場合の最適利得}}$

アウトライン

かな漢字変換



- 変換候補をどのような順番で表示すれば効率よく変換できるか？
- 上の例の場合「降下」と変換するには 4 回リストを辿ることに

リストアクセス問題

- 初期リスト $(e_1, e_2, \ldots, e_n)$
- 各ステップ $t = 1, 2, \ldots$ の動作
  - リクエスト：探索する要素 $r_t \in \{e_1, \ldots, e_n\}$
  - 前から順に r_t を探索
  - 必要ならば要素を並べ替え
- 目標：minimize $\sum_t$(探索コスト$_t$ + 移動コスト$_t$)

並べ替えで許される操作とコスト

- r_t を前方の任意の位置に移動（コスト 0）
- r_t 以外の隣り合う 2 つの要素を入れ替え（コスト 1）

リストアクセス問題の例

- 初期リスト：(e_1, e_2, e_3)
- リクエスト列：$e_1, e_1, e_2, e_2, e_3, e_2$

リクエスト列が既知の場合，最適な並べ替え方は

1. (e_1, e_2, e_3): e_1 の探索コスト 1
2. (e_1, e_2, e_3): e_1 の探索コスト 1
3. (e_1, e_2, e_3): e_2 の探索コスト 2，移動コスト 0 で e_2 を先頭に移動
4. (e_2, e_1, e_3): e_2 の探索コスト 1
5. (e_2, e_1, e_3): e_3 の探索コスト 3
6. (e_2, e_1, e_3): e_2 の探索コスト 1

コストの合計 $= 9$

使えそうなアルゴリズム

- No-Action (NOA): 並べ替えをしない

$$(e_1, e_2, e_3, e_4) \xrightarrow{\text{request } e_3} (e_1, e_2, e_3, e_4)$$

- Transpose (TR): 検索したデータを 1 つ前と入れ替える

$$(e_1, e_2, e_3, e_4) \xrightarrow{\text{request } e_3} (e_1, e_3, e_2, e_4)$$

- Move-To-Front (MTF): 検索したデータを先頭に移動する

$$(e_1, e_2, e_3, e_4) \xrightarrow{\text{request } e_3} (e_3, e_1, e_2, e_4)$$

- Frequency Count (FC): 過去のアクセス数の大きい順に並べる

$$\underset{5}{(e_1}, \underset{3}{e_2}, \underset{3}{e_3}, \underset{2}{e_4}) \xrightarrow{\text{request } e_3} \underset{5}{(e_1}, \underset{4}{e_3}, \underset{3}{e_2}, \underset{2}{e_4})$$

どれが良い？

NOA（並べ替えをしない）

e_n が T 回リクエストされた場合 ⟶ 競合比 $\geq \frac{nT}{n+T-1} \xrightarrow{T\to\infty} n$

- NOA アルゴリズム ⟶ コストは nT
- 最適な方法 ⟶ コストは $n + T - 1$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭なので探索コスト 1

どのような並べ方でも各ステップでの探索コストは 1 以上であり，
NOA アルゴリズムの各ステップでのコストは n 以下
⟶ 競合比 $\leq n$

定理
NOA の競合比は n

NOA（並べ替えをしない）

e_n が T 回リクエストされた場合 ⟶ 競合比 $\geq \frac{nT}{n+T-1} \xrightarrow{T\to\infty} n$

- NOA アルゴリズム ⟶ コストは nT
- 最適な方法 ⟶ コストは $n + T - 1$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭なので探索コスト 1

どのような並べ方でも各ステップでの探索コストは 1 以上であり，
NOA アルゴリズムの各ステップでのコストは n 以下
⟶ 競合比 $\leq n$

定理
NOA の競合比は n

NOA（並べ替えをしない）

e_n が T 回リクエストされた場合 ⟶ 競合比 $\geq \frac{nT}{n+T-1} \xrightarrow{T\to\infty} n$

- NOA アルゴリズム ⟶ コストは nT
- 最適な方法 ⟶ コストは $n + T - 1$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭なので探索コスト 1

どのような並べ方でも各ステップでの探索コストは 1 以上であり，
NOA アルゴリズムの各ステップでのコストは n 以下
⟶ 競合比 $\leq n$

定理

NOA の競合比は n

NOA（並べ替えをしない）

e_n が T 回リクエストされた場合 ⟶ 競合比 $\geq \frac{nT}{n+T-1} \xrightarrow{T\to\infty} n$

- NOA アルゴリズム ⟶ コストは nT
- 最適な方法 ⟶ コストは $n + T - 1$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭なので探索コスト 1

どのような並べ方でも各ステップでの探索コストは 1 以上であり，
NOA アルゴリズムの各ステップでのコストは n 以下
⟶ 競合比 $\leq n$

定理

NOA の競合比は n

TR（一つ前と入れ替え）

e_n と e_{n-1} を交互に T 回リクエスト ⟶ 競合比 $\geq \frac{2nT}{2n+3(T-1)} \xrightarrow{T\to\infty} \frac{2n}{3}$

- TR アルゴリズム ⟶ コストは $2nT$
- 最適な方法 ⟶ コストは $2n + 3(T-1)$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 次の探索で e_{n-1} を 2 番目に移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭，e_{n-1} は 2 番目なので探索コストは 1 と 2

NOA と同様の解析により，競合比は n 以下であることが言える

定理

TR の競合比は $\Theta(n)$

TR（一つ前と入れ替え）

e_n と e_{n-1} を交互に T 回リクエスト ⟶ 競合比 $\geq \frac{2nT}{2n+3(T-1)} \xrightarrow{T\to\infty} \frac{2n}{3}$

- TR アルゴリズム ⟶ コストは $2nT$
- 最適な方法 ⟶ コストは $2n + 3(T-1)$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 次の探索で e_{n-1} を 2 番目に移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭，e_{n-1} は 2 番目なので探索コストは 1 と 2

NOA と同様の解析により，競合比は n 以下であることが言える

定理

TR の競合比は $\Theta(n)$

TR（一つ前と入れ替え）

e_n と e_{n-1} を交互に T 回リクエスト ⟶ 競合比 $\geq \frac{2nT}{2n+3(T-1)} \xrightarrow{T\to\infty} \frac{2n}{3}$

- TR アルゴリズム ⟶ コストは $2nT$
- 最適な方法 ⟶ コストは $2n + 3(T - 1)$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 次の探索で e_{n-1} を 2 番目に移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭，e_{n-1} は 2 番目なので探索コストは 1 と 2

NOA と同様の解析により，競合比は n 以下であることが言える

定理

TR の競合比は $\Theta(n)$

TR（一つ前と入れ替え）

e_n と e_{n-1} を交互に T 回リクエスト ⟶ 競合比 $\geq \frac{2nT}{2n+3(T-1)} \xrightarrow{T\to\infty} \frac{2n}{3}$

- TR アルゴリズム ⟶ コストは $2nT$
- 最適な方法 ⟶ コストは $2n + 3(T - 1)$
  - 最初の探索で e_n を先頭へ移す（探索コスト n，移動コスト 0）
  - 次の探索で e_{n-1} を 2 番目に移す（探索コスト n，移動コスト 0）
  - 以降の探索では e_n は先頭，e_{n-1} は 2 番目なので探索コストは 1 と 2

NOA と同様の解析により，競合比は n 以下であることが言える

定理

TR の競合比は $\Theta(n)$

MTF（先頭へ移動）の下界

最後の要素を n 回リクエスト ⟶ 競合比 $\geq \frac{n^2}{n(n+1)/2} = \frac{2n}{n+1}$

- MTF アルゴリズム ⟶ コストは n^2
  $(e_1, e_2, e_3, e_4) \to (e_4, e_1, e_2, e_3) \to (e_3, e_4, e_1, e_2) \to (e_2, e_3, e_4, e_1)$
- NOA アルゴリズム ⟶ コストは $n(n+1)/2$
- 最適な方法 ⟶ コストは $n(n+1)/2$ 以下

上界は？ ⟶ 2 以下であることを示す

MTF（先頭へ移動）の下界

最後の要素を n 回リクエスト ⟶ 競合比 $\geq \frac{n^2}{n(n+1)/2} = \frac{2n}{n+1}$

- MTF アルゴリズム ⟶ コストは n^2
  $(e_1, e_2, e_3, e_4) \to (e_4, e_1, e_2, e_3) \to (e_3, e_4, e_1, e_2) \to (e_2, e_3, e_4, e_1)$
- NOA アルゴリズム ⟶ コストは $n(n+1)/2$
- 最適な方法 ⟶ コストは $n(n+1)/2$ 以下

上界は？ ⟶ 2 以下であることを示す

MTF（先頭へ移動）の下界

最後の要素を n 回リクエスト ⟶ 競合比 $\geq \frac{n^2}{n(n+1)/2} = \frac{2n}{n+1}$

- MTF アルゴリズム ⟶ コストは n^2
  $(e_1, e_2, e_3, e_4) \to (e_4, e_1, e_2, e_3) \to (e_3, e_4, e_1, e_2) \to (e_2, e_3, e_4, e_1)$
- NOA アルゴリズム ⟶ コストは $n(n+1)/2$
- 最適な方法 ⟶ コストは $n(n+1)/2$ 以下

上界は？ ⟶ 2 以下であることを示す

ポテンシャル関数

順序が異なるペアの数

Φ_t: t 番目のリクエストを処理後の MTF と OPT のリストの間での転倒数

例

t	0	1	2	3	$\cdots$
r_t	—	e_3	e_2	e_3	$\cdots$
MTF	(e_1, e_2, e_3, e_4)	(e_3, e_1, e_2, e_4)	(e_2, e_3, e_1, e_4)	(e_3, e_2, e_1, e_4)	$\cdots$
OPT	(e_1, e_2, e_3, e_4)	(e_2, e_3, e_1, e_4)	(e_2, e_3, e_1, e_4)	(e_2, e_3, e_1, e_4)	$\cdots$
Φ_t	0	2	0	1	$\cdots$

考察

- $\Phi_0 = 0$
- $0 \leq \Phi_t \leq n(n+1)/2$ $(\forall t)$

償却コスト

- MTF_t: MTF の t 番目のリクエストに対する探索コスト
- $\mathrm{MTF}_t^* = \mathrm{MTF}_t + \Phi_t - \Phi_{t-1}$: 償却コスト
- OPT_t: 最適アルゴリズムの t 番目のリクエストに対するコスト

考察

$$\sum_{t=1}^{T} \mathrm{MTF}_t^* = \sum_{t=1}^{T} \mathrm{MTF}_t + \underbrace{\Phi_T}_{\geq 0} - \underbrace{\Phi_0}_{=0} \geq \sum_{t=1}^{T} \mathrm{MTF}_t$$

$\longrightarrow$ $\mathrm{MTF}_t^* \leq \alpha \cdot \mathrm{OPT}_t$ ならば競合比 α 以下が示せる！

ある t では MTF_t と OPT_t の比は大きくなるかもしれないので，ポテンシャルを用いてコストを「ならす」

償却コスト

- MTF_t: MTF の t 番目のリクエストに対する探索コスト
- $\mathrm{MTF}_t^* = \mathrm{MTF}_t + \Phi_t - \Phi_{t-1}$: 償却コスト
- OPT_t: 最適アルゴリズムの t 番目のリクエストに対するコスト

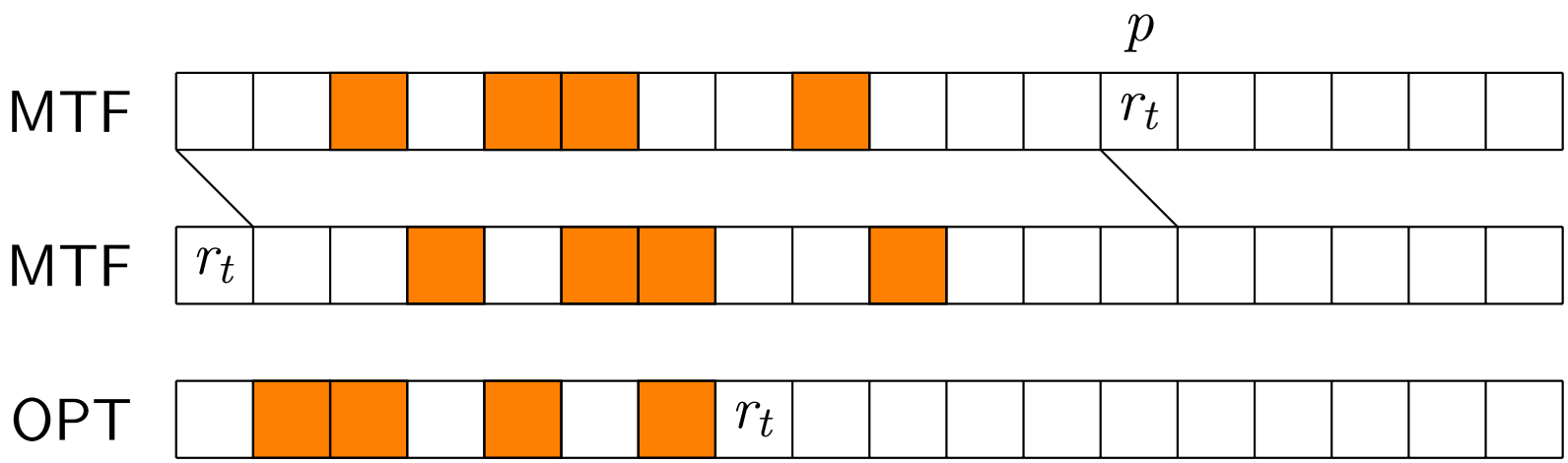
考察

$$\sum_{t=1}^{T} \mathrm{MTF}_t^* = \sum_{t=1}^{T} \mathrm{MTF}_t + \underbrace{\Phi_T}_{\geq 0} - \underbrace{\Phi_0}_{=0} \geq \sum_{t=1}^{T} \mathrm{MTF}_t$$

→ $\mathrm{MTF}_t^* \leq \alpha \cdot \mathrm{OPT}_t$ ならば競合比 α 以下が示せる！

ある t では MTF_t と OPT_t の比は大きくなるかもしれないので，ポテンシャルを用いてコストを「ならす」

MTF の競合比



MTF と OPT の両方で r_t の前にある要素 (q 個)

探索コストの下限　移動コスト

- $\mathrm{MTF}_t = p,\ \ \mathrm{OPT}_t \geq q+1 + r$
- $\Phi_t - \Phi_{t-1} \leq 2q - p + 1 + r$
  - MTF の要素移動によるポテンシャル変化は $2q - p + 1$ (q 増えて $p - q - 1$ 減る)
  - OPT の要素移動によるポテンシャルの増加は高々 r (r_t の移動では増加しない)

r_t に対し MTF では前，OPT では後のもの

→ $\mathrm{MTF}^*_t = \mathrm{MTF}_t + \Phi_t - \Phi_{t-1} \leq 2q + 1 + r \leq 2\mathrm{OPT}_t - 1$

→ MTF の競合比は 2 以下　-1 を使うと $2n/(n+1)$ 以下も言える

リストアクセス問題 まとめ

- No-Action (NOA): 並べ替えをしない
  → 競合比 n
- Transpose (TR): 検索したデータを 1 つ前と入れ替える
  → 競合比 $\Theta(n)$
- Move-To-Front (MTF): 検索したデータを先頭に移動する
  → 競合比 $\frac{2n}{n+1}$
- Frequency Count (FC): 過去のアクセス数の大きい順に並べる
  → 競合比 $\Theta(n)$　(演習)

MTF より良いアルゴリズムはある？ → ない！

リストアクセス問題 まとめ

- No-Action (NOA): 並べ替えをしない
  → 競合比 n
- Transpose (TR): 検索したデータを 1 つ前と入れ替える
  → 競合比 $\Theta(n)$
- Move-To-Front (MTF): 検索したデータを先頭に移動する
  → 競合比 $\frac{2n}{n+1}$
- Frequency Count (FC): 過去のアクセス数の大きい順に並べる
  → 競合比 $\Theta(n)$　(演習)

MTF より良いアルゴリズムはある？ → ない！

リストアクセス問題の競合比の下界

定理

任意の[オンラインアルゴリズム](ALG)の競合比は $\frac{2n}{n+1}$ 以上

証明: ALG に対して，最後の要素を T 回リクエストされた時を考える

- ALG のコストは nT
- リストの変更は最初だけのアルゴリズム $n!$ 通りについて
  - 最初のリスト変更によるコストの合計 $\leq n^2 \cdot n!$
  - 各リクエストの探索コストの合計 $\leq \sum_{k=1}^{n} k \cdot (n-1)! = \frac{n+1}{2} \cdot n!$
  - 総コストの合計 $\leq (n^2 + \frac{n+1}{2} \cdot T)n!$

  → どれかのアルゴリズムのコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下
- 最適な方法のコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

競合比は $\frac{nT}{n^2+\frac{n+1}{2} \cdot T} \xrightarrow{T \to \infty} \frac{2n}{n+1}$ 以上

リストアクセス問題の競合比の下界

定理 (ALG)

任意のオンラインアルゴリズムの競合比は $\frac{2n}{n+1}$ 以上

証明: ALG に対して，最後の要素を T 回リクエストされた時を考える

- ALG のコストは nT
- リストの変更は最初だけのアルゴリズム $n!$ 通りについて
  - 最初のリスト変更によるコストの合計 $\leq n^2 \cdot n!$
  - 各リクエストの探索コストの合計 $\leq \sum_{k=1}^{n} k \cdot (n-1)! = \frac{n+1}{2} \cdot n!$
  - 総コストの合計 $\leq (n^2 + \frac{n+1}{2} \cdot T)n!$

  → どれかのアルゴリズムのコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下
- 最適な方法のコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

競合比は $\frac{nT}{n^2 + \frac{n+1}{2} \cdot T} \xrightarrow{T \to \infty} \frac{2n}{n+1}$ 以上

リストアクセス問題の競合比の下界

定理 ALG

任意のオンラインアルゴリズムの競合比は $\frac{2n}{n+1}$ 以上

証明: ALG に対して，最後の要素を T 回リクエストされた時を考える

- ALG のコストは nT
- リストの変更は最初だけのアルゴリズム $n!$ 通りについて
  - 最初のリスト変更によるコストの合計 $\leq n^2 \cdot n!$
  - 各リクエストの探索コストの合計 $\leq \sum_{k=1}^{n} k \cdot (n-1)! = \frac{n+1}{2} \cdot n!$
  - 総コストの合計 $\leq (n^2 + \frac{n+1}{2} \cdot T)n!$

→ どれかのアルゴリズムのコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

- 最適な方法のコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

競合比は $\frac{nT}{n^2 + \frac{n+1}{2} \cdot T} \xrightarrow{T \to \infty} \frac{2n}{n+1}$ 以上

リストアクセス問題の競合比の下界

定理 (ALG)

任意のオンラインアルゴリズムの競合比は $\frac{2n}{n+1}$ 以上

証明: ALG に対して，最後の要素を T 回リクエストされた時を考える

- ALG のコストは nT
- リストの変更は最初だけのアルゴリズム $n!$ 通りについて
  - 最初のリスト変更によるコストの合計 $\leq n^2 \cdot n!$
  - 各リクエストの探索コストの合計 $\leq \sum_{k=1}^{n} k \cdot (n-1)! = \frac{n+1}{2} \cdot n!$
  - 総コストの合計 $\leq (n^2 + \frac{n+1}{2} \cdot T)n!$

  → どれかのアルゴリズムのコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下
- 最適な方法のコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

競合比は $\frac{nT}{n^2 + \frac{n+1}{2} \cdot T} \xrightarrow{T \to \infty} \frac{2n}{n+1}$ 以上

リストアクセス問題の競合比の下界

定理

任意の[オンラインアルゴリズム](ALG)の競合比は $\frac{2n}{n+1}$ 以上

証明: ALG に対して，最後の要素を T 回リクエストされた時を考える

- ALG のコストは nT
- リストの変更は最初だけのアルゴリズム $n!$ 通りについて
  - 最初のリスト変更によるコストの合計 $\leq n^2 \cdot n!$
  - 各リクエストの探索コストの合計 $\leq \sum_{k=1}^{n} k \cdot (n-1)! = \frac{n+1}{2} \cdot n!$
  - 総コストの合計 $\leq (n^2 + \frac{n+1}{2} \cdot T)n!$

  → どれかのアルゴリズムのコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下
- 最適な方法のコストは $n^2 + \frac{n+1}{2} \cdot T$ 以下

競合比は $\frac{nT}{n^2 + \frac{n+1}{2} \cdot T} \xrightarrow{T \to \infty} \frac{2n}{n+1}$ 以上

アウトライン

まとめ

- オンラインアルゴリズムの性能指標として競合比を定義
- スキーレンタル問題を解析
- リストアクセス問題を解析

演習

リストアクセス問題に対し，「過去のアクセス数の大きい順に並べる」という *FC* アルゴリズムの競合比が $\Theta(n)$ となることを示せ