オンラインナップサック問題に対するアルゴリズム

河瀬康志 (東京大学)

2020 年 11 月 14 日
OR **学会関西支部**

自己紹介: 河瀬 康志 (かわせ やすし)

経歴

- **出身: 大阪・吹田**
- **〜2014 年 3 月　東京大学大学院情報理工学研究科 博士課程**
- 2014 年 4 月〜2020 年 9 月　東京工業大学工学部社会工学科 助教
  (2016 年 4 月より学科再編により工学院経営工学系に異動)
- 2020 年 10 月〜 東京大学大学院情報理工学研究科 特任准教授

専門分野

- **オンライン最適化**
- **アルゴリズム的ゲーム理論**
- **離散数学**

自己紹介: 河瀬 康志 (かわせ やすし)

経歴

- 出身: 大阪・吹田
- ～2014 年 3 月　東京大学大学院情報理工学研究科 博士課程
- 2014 年 4 月～2020 年 9 月　東京工業大学工学部社会工学科 助教
  (2016 年 4 月より学科再編により工学院経営工学系に異動)
- 2020 年 10 月～ 東京大学大学院情報理工学研究科 特任准教授

専門分野

- オンライン最適化
- アルゴリズム的ゲーム理論
- 離散数学

アウトライン

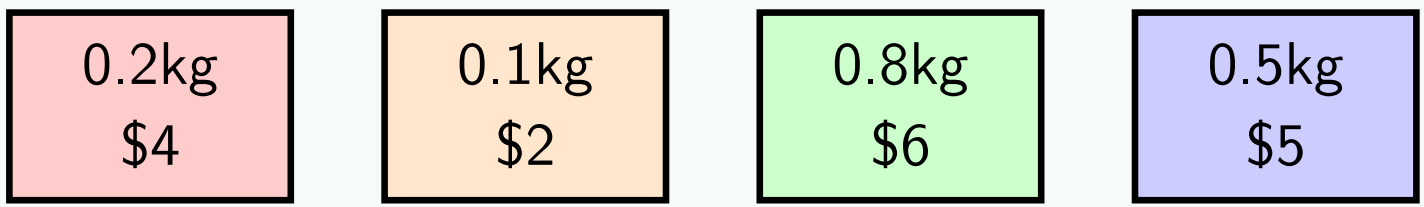
1. オンラインナップサック問題
2. 除去可能設定
3. 資源拡大モデルとバッファモデル
4. まとめ

オフラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例



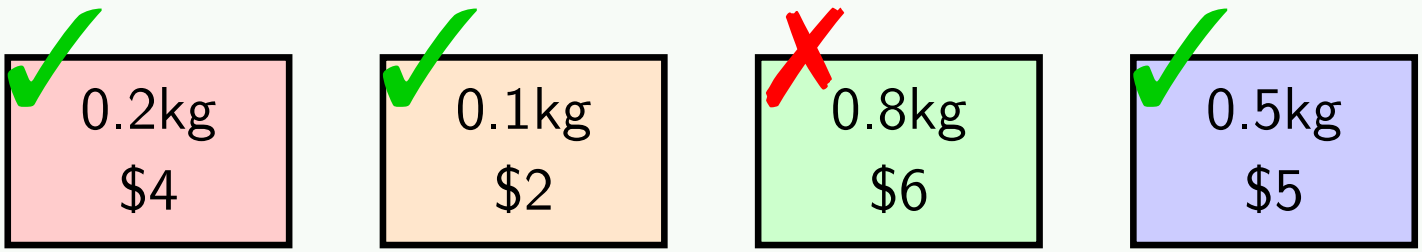
オフラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

最適値 $11



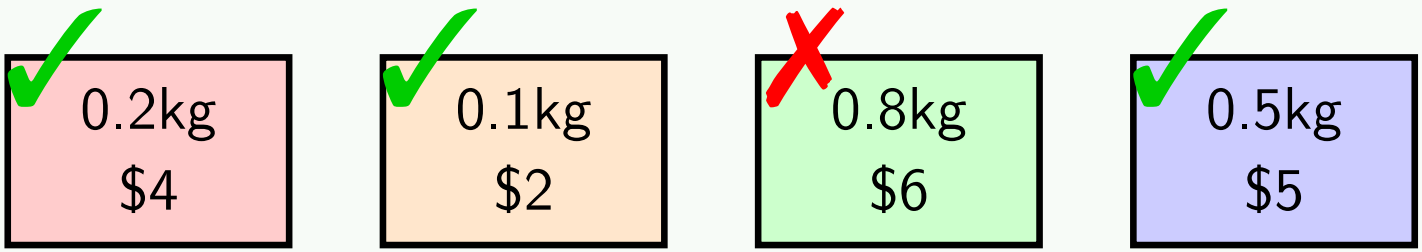
オフラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

最適値 $11



- 最も基本的な組合せ最適化問題の 1 つ
- 様々な応用: 予算内で購入する物資，制限時間内に処理するプロジェクト
- NP 困難だが効率的に近似解を求めることは可能

オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

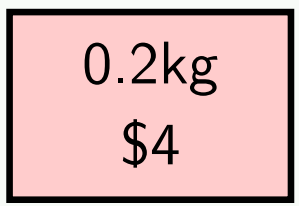
例

オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例



オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

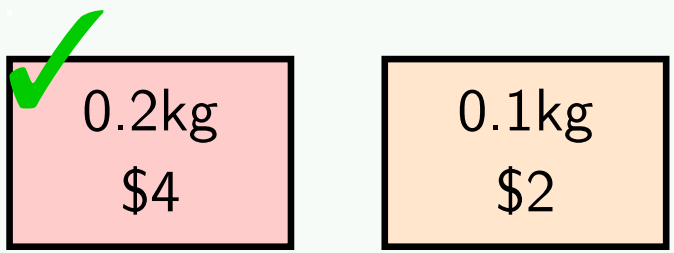
例

オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

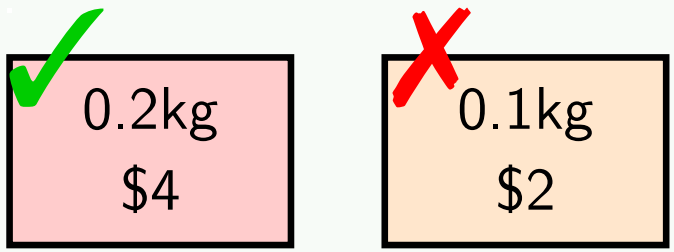


オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

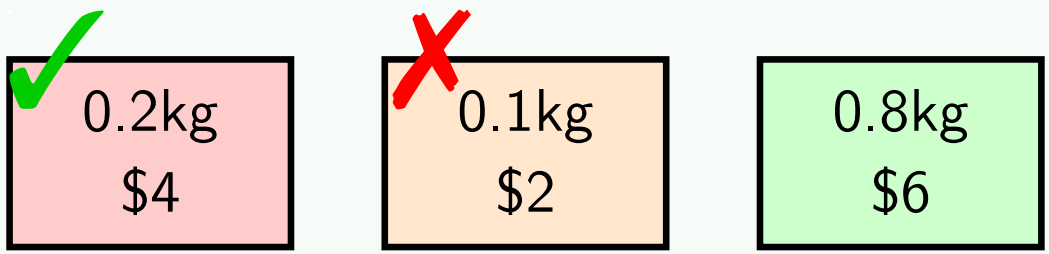


オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

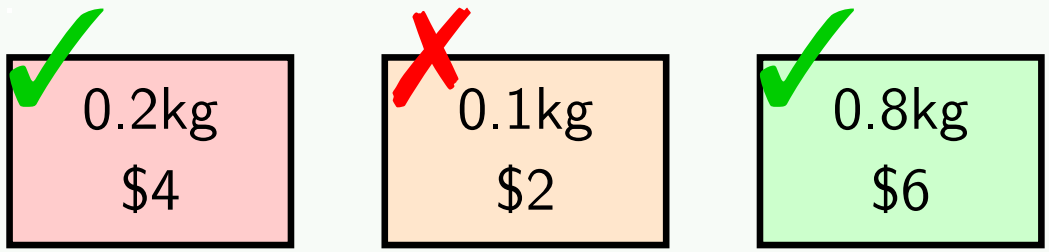


オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

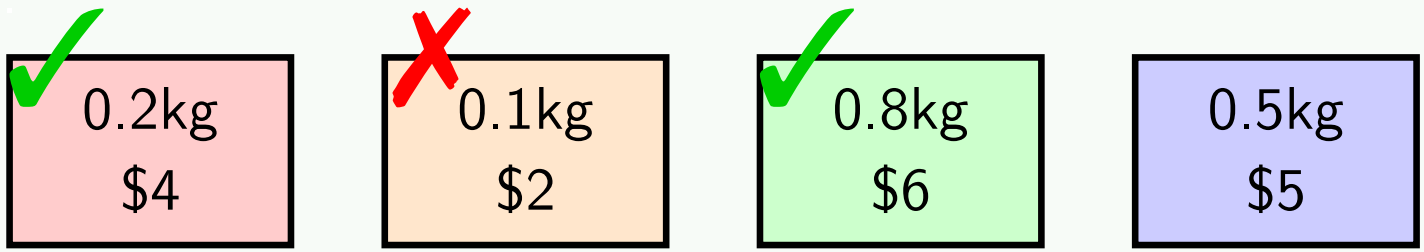


オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

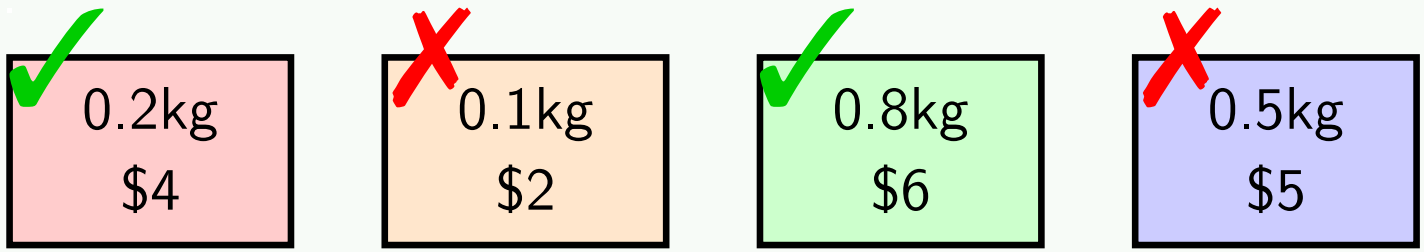


オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例



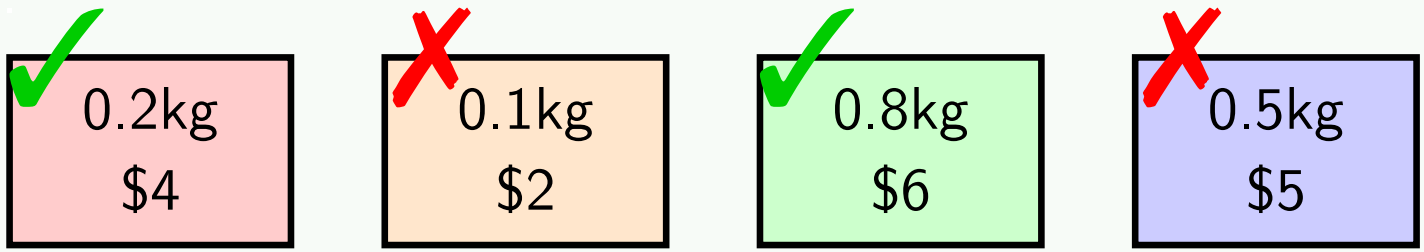
オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

利得: $10



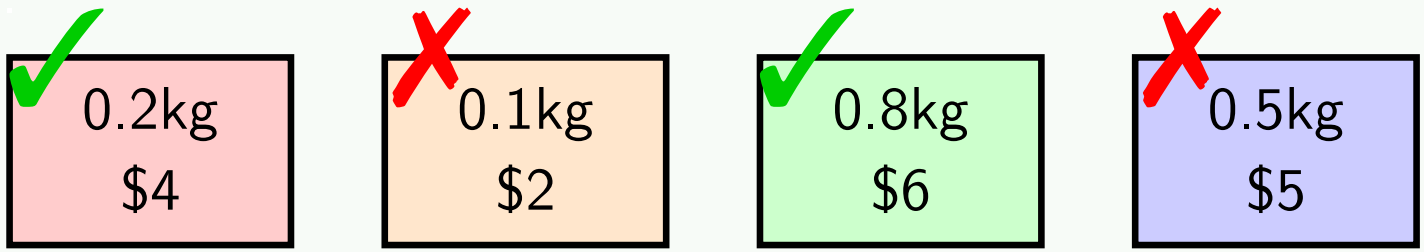
オンラインナップサック問題

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

利得: $10



- 最も基本的な組合せ最適化問題の 1 つ
- 様々な応用: 物資やプロジェクトが逐次的に来る状況
- よい解を得られる？？？

オフライン vs オンライン

オフライン最適化

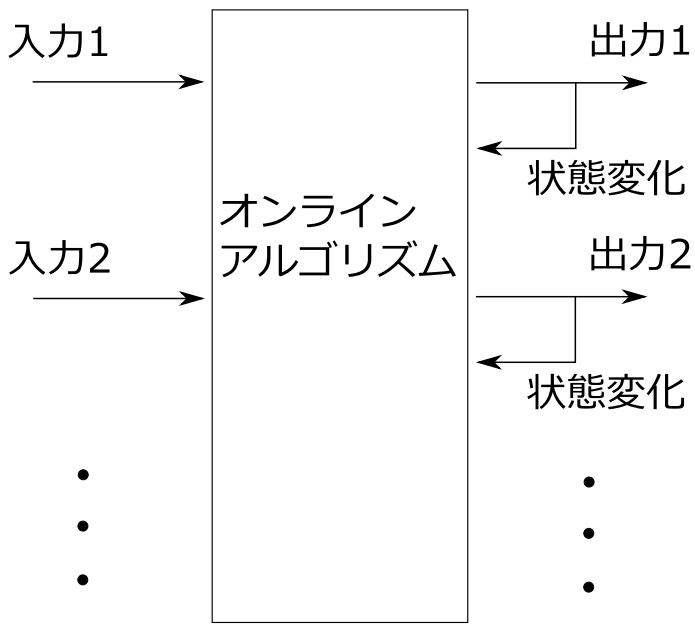
入力 **一度**に与えられる

目的 “よい” 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 将来を見越した “よい” 選択



(最大化問題に対する) 競合比

$\mathrm{OPT}(I)$ 入力 I に対するオフラインでの最適利得
$\mathrm{ALG}(I)$ 入力 I に対するオンラインアルゴリズム ALG での利得

$$\mathrm{ALG}\text{ の }I\text{ に対する競合比} = \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

- 競合比の値は少なくとも 1
- 小さいほど嬉しい
- 乱択アルゴリズムを用いる場合は定義がいくつかある
  oblivious, adaptive-online, adaptive-offline

(最大化問題に対する) 競合比

$\mathrm{OPT}(I)$ 入力 I に対するオフラインでの最適利得
$\mathrm{ALG}(I)$ 入力 I に対するオンラインアルゴリズム ALG での利得

$$\mathrm{ALG}\text{ の }I\text{ に対する競合比} = \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

$$\mathrm{ALG}\text{ の競合比} = \sup_{I} \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

- 競合比の値は少なくとも 1
- 小さいほど嬉しい
- 乱択アルゴリズムを用いる場合は定義がいくつかある
  oblivious, adaptive-online, adaptive-offline

(最大化問題に対する) 競合比

$\mathrm{OPT}(I)$ 入力 I に対するオフラインでの最適利得
$\mathrm{ALG}(I)$ 入力 I に対するオンラインアルゴリズム ALG での利得

$$\mathrm{ALG}\text{ の }I\text{ に対する競合比} = \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

$$\mathrm{ALG}\text{ の競合比} = \sup_{I} \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

$$\text{競合比} = \inf_{\mathrm{ALG}} \sup_{I} \frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)}$$

- 競合比の値は少なくとも 1
- 小さいほど嬉しい
- 乱択アルゴリズムを用いる場合は定義がいくつかある
  oblivious, adaptive-online, adaptive-offline

オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

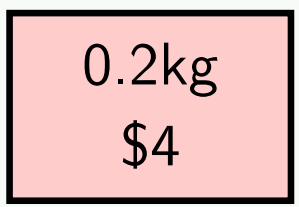
例

オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例



オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

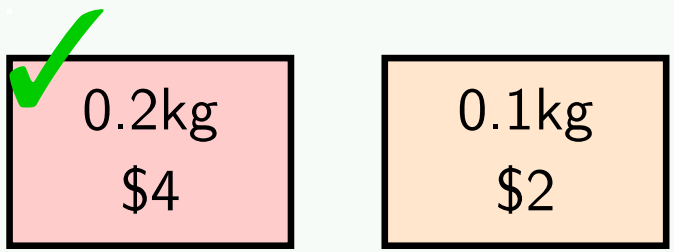
例

オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

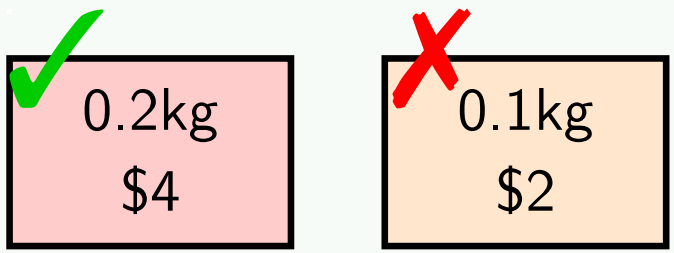


オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

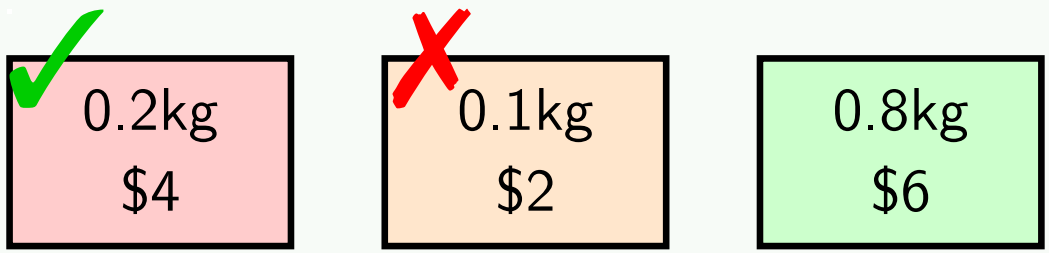


オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

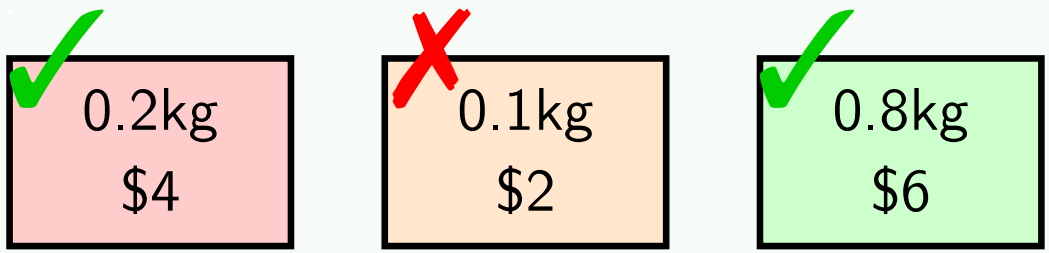


オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

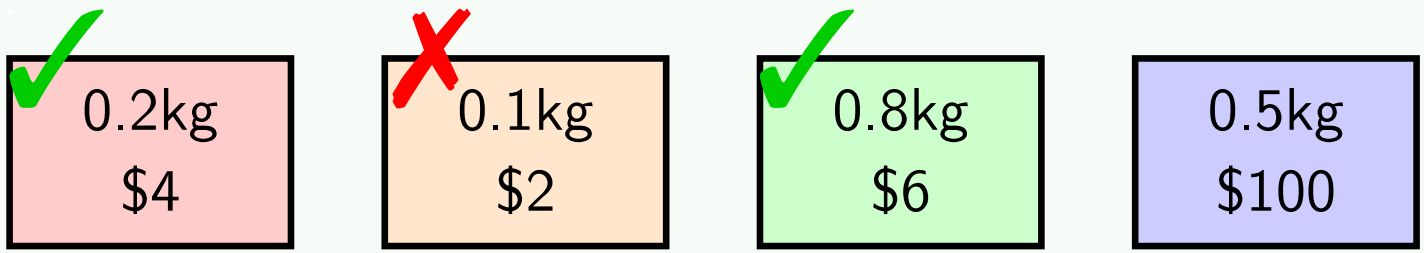


オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

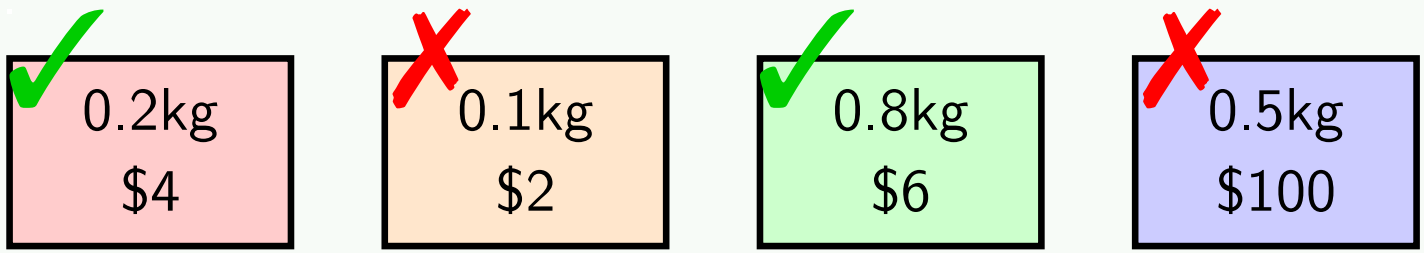


オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例



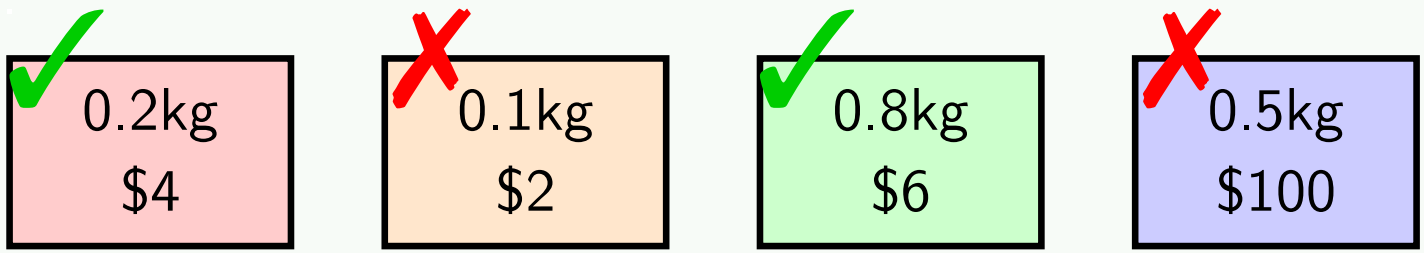
オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

競合比: 106/10



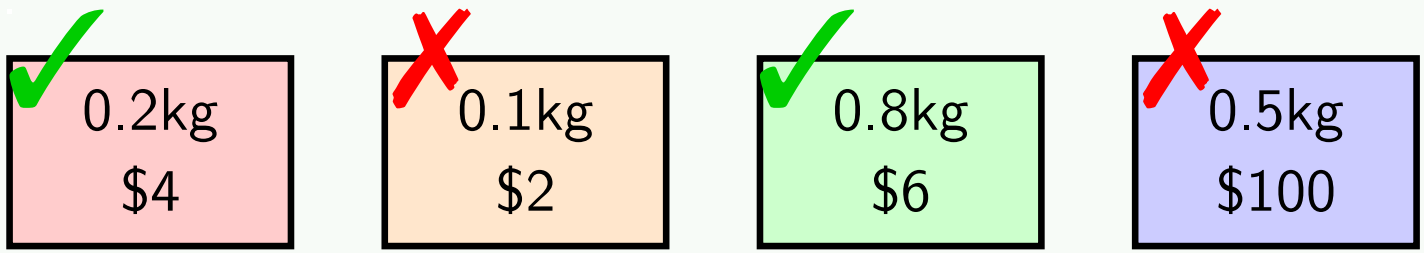
オンラインナップサック問題の競合比

入力: 容量 1kg のナップサック
それぞれ重さと価値をもつアイテムの集合 (逐次的に与えられる)

目的: 価値の和が大きくなるようナップサックにアイテムを詰める

例

競合比: 106/10



定理 [Marchetti-Spaccamela and Vercellis 1995]

オンラインナップサック問題の競合比は無限大

$\rightsquigarrow$ **適用したい状況に合わせた適切な仮定が必要**

様々な設定

- **除去可能** (removable, disposable, preemptive)

  一度選んだアイテムを後から捨てることができる

- **入力されるアイテムに仮定**
  - **価値が重さに比例**
  - **価値が全て同じ (要素数最大化)**
  - **重さが全て同じ (要素数制約)**
  - **重さが全て小さい**
  - **価値，重さ，密度 (単位重さあたりの価値) に上下限**
- **アドバイスを利用できる**
- **資源増大** (resource augmentation)

  アルゴリズムは容量 R (≥ 1) **のナップサックを使える**

- **入力の分布に仮定**
  - **確率分布に従ってアイテムは与えられる**
  - **ランダムな順序でアイテムは与えられる**

不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]

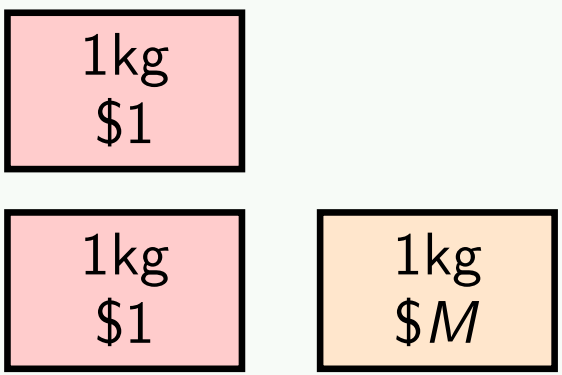
不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]



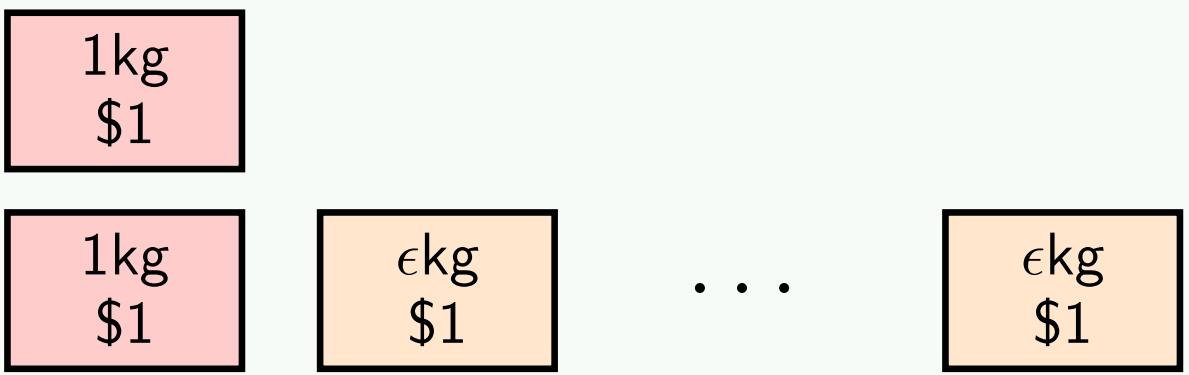
不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]



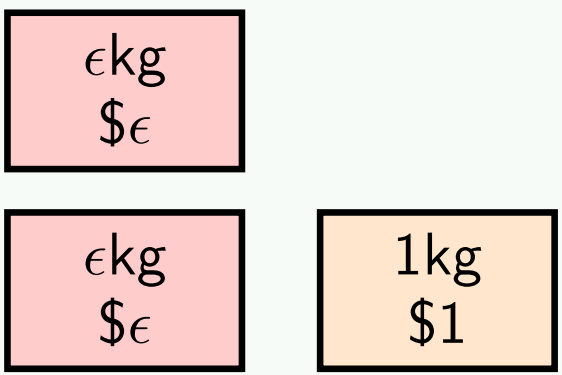
不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]



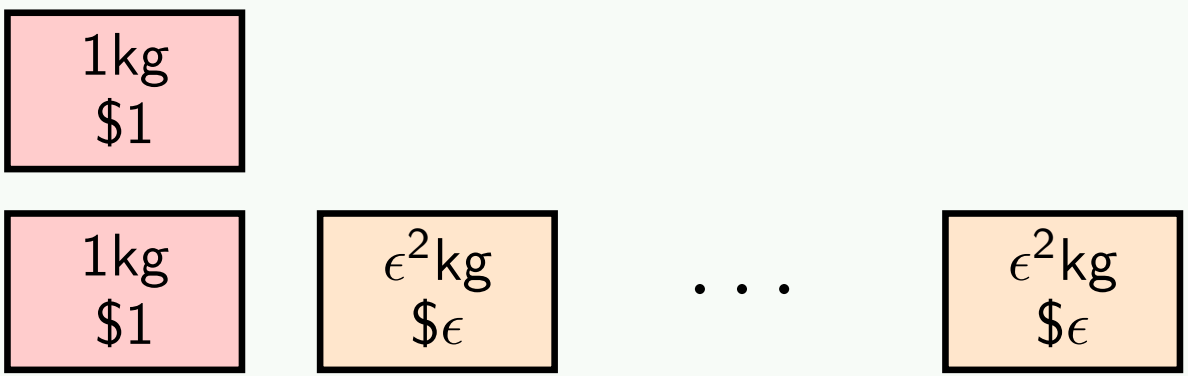
不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]



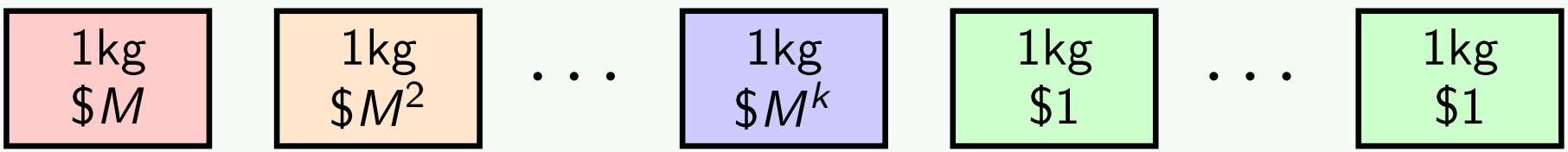
不可能性

残念ながら，多くの設定において競合比は無限大になる

定理

オンラインナップサック問題の競合比は下記の限定された状況でも無限大

- **アイテムの重さが全て** 1 [Marchetti-Spaccamela and Vercellis 1995]
- **アイテムの価値が全て** 1 folklore
- **密度が全て** 1 folklore
- **除去可能** [Iwama and Zhang 2007]
- **アイテム数** n **で** $\log_2 n$ **ビット未満のアドバイス** [Böckenhauer et al. 2014]



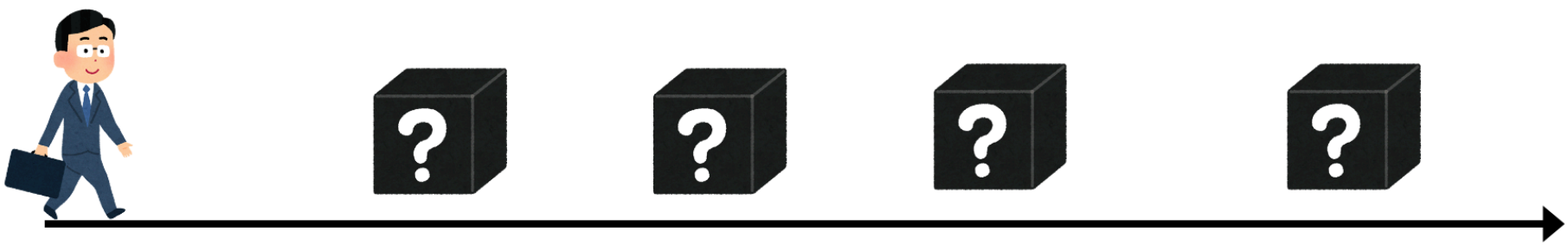
アウトライン

1. オンラインナップサック問題
2. **除去可能設定**
3. 資源拡大モデルとバッファモデル
4. まとめ

除去可能

以後は除去可能な場合を考える

- 重量制限を満たすようにアイテムを選択して運ぶ
- キャンセル (返金手続き) 可能な契約
- ストリーミングアルゴリズムへの応用
- 安定マッチング問題への応用 (受入保留方式)



重さ一定

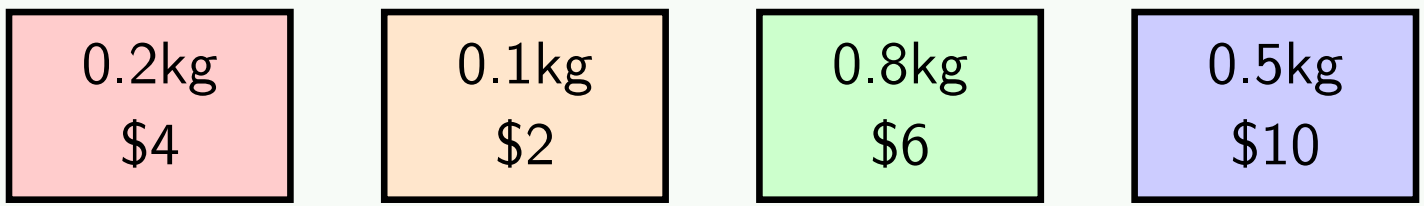
価値が高い物を優先して貪欲にアイテムをキープ

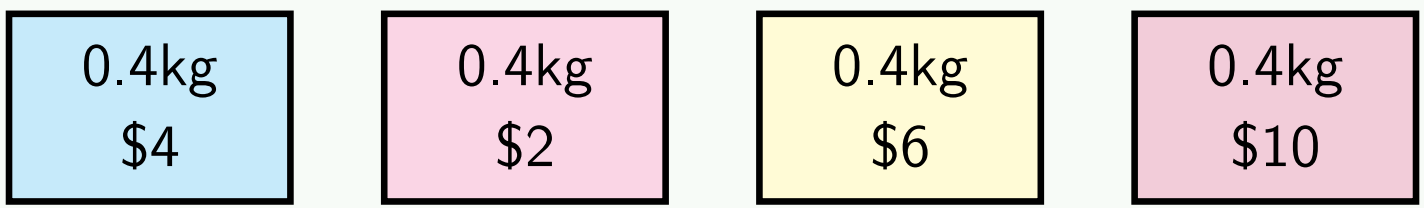
定理

除去可能オンラインナップサック問題では最大価値のアイテムをとれる

定理

アイテムの重さが全て同じ場合，最適解をとれる (競合比 1)





価値一定

密度最大の中で最小のアイテムを優先して貪欲にアイテムをキープ

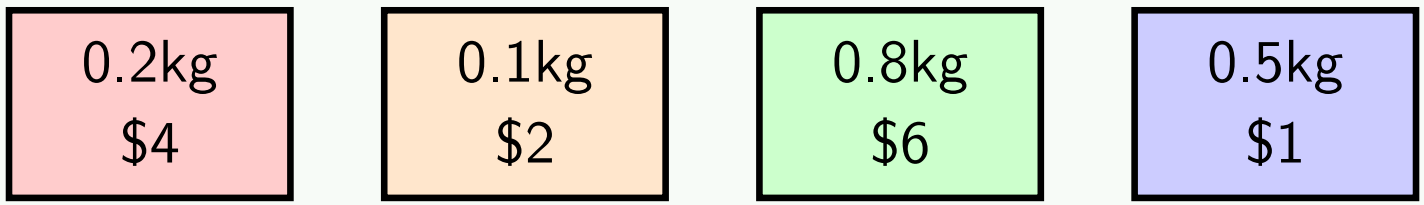
定理

密度最大の中で最小のアイテムを優先してあふれる直前まで取った解

除去可能オンラインナップサック問題では**貪欲解**を (含む解を) とれる

定理

アイテムの価値が全て同じ場合，最適解をとれる (競合比 1)



どのような順でアイテムがきても，貪欲解のアイテムは捨てられない

価値と重さの両方が同じアイテムは入れ替わり得る

あふれたアイテムの後に取るアイテムはどうなるかわからない

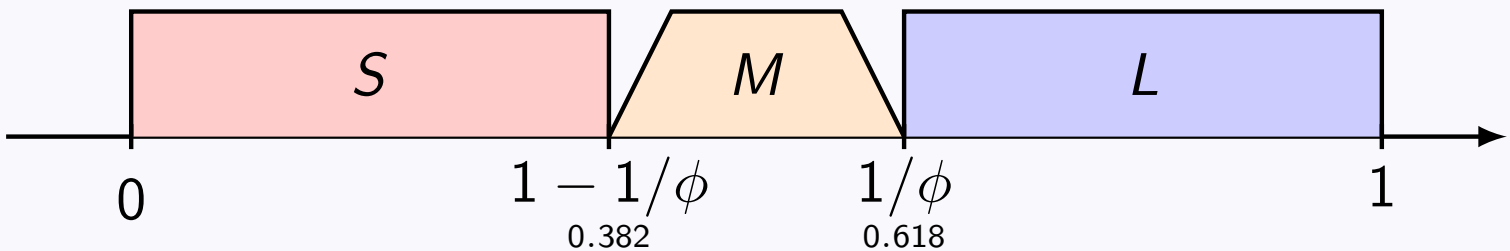
密度一定

競合比 $\phi = \frac{1+\sqrt{5}}{2}$ のアルゴリズム [Iwama and Taketomi 2005]

現状保持しているアイテムと新しいアイテムを，以下の順で貪欲選択

1. **L アイテムについて大きい物を優先**
2. **M アイテムについて小さいものを優先**
3. **S アイテムについて大きい物を優先**

$1/\phi$ 以上の解を得られたら以後は解を変更しない



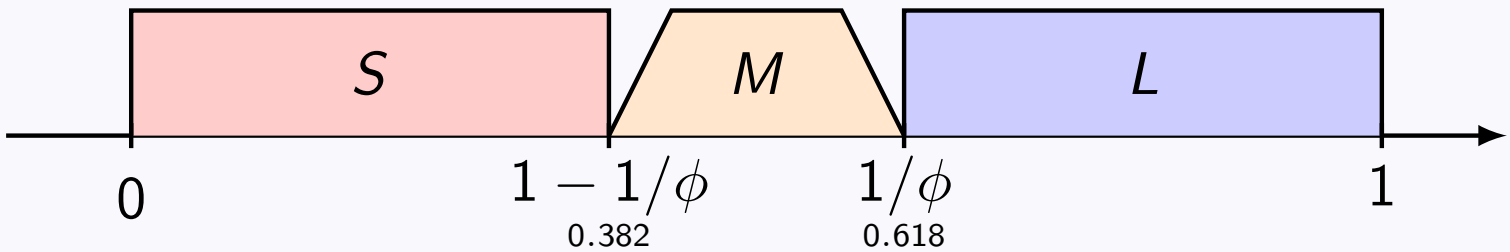
密度一定

競合比 $\phi = \frac{1+\sqrt{5}}{2}$ のアルゴリズム [Iwama and Taketomi 2005]

現状保持しているアイテムと新しいアイテムを，以下の順で貪欲選択

1. L アイテムについて大きい物を優先
2. M アイテムについて小さいものを優先
3. S アイテムについて大きい物を優先

$1/\phi$ 以上の解を得られたら以後は解を変更しない



略証

- L アイテム有り $\Rightarrow$ 競合比 $\leq \frac{1}{1/\phi} = \phi = \frac{1+\sqrt{5}}{2}$
- M アイテムを 2 個とれる $\Rightarrow$ 競合比 $\leq \frac{1}{2(1-1/\phi)} < \frac{1+\sqrt{5}}{2}$
- S アイテムを全部とれない $\Rightarrow$ 競合比 $\leq \frac{1}{1-(1-1/\phi)} = \frac{1+\sqrt{5}}{2}$

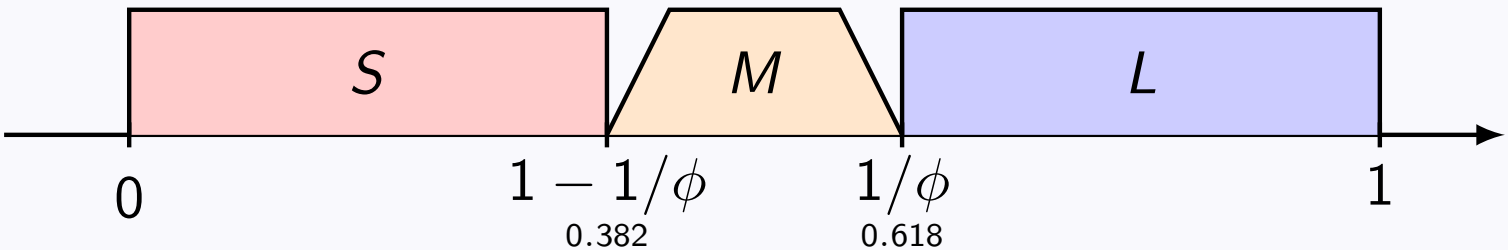
密度一定

競合比 $\phi = \frac{1+\sqrt{5}}{2}$ のアルゴリズム [Iwama and Taketomi 2005]

現状保持しているアイテムと新しいアイテムを，以下の順で貪欲選択

1. *L* アイテムについて大きい物を優先
2. *M* アイテムについて小さいものを優先
3. *S* アイテムについて大きい物を優先

$1/\phi$ 以上の解を得られたら以後は解を変更しない



略証

- L アイテムなし，M アイテムを 1 個まで，S アイテム全部とれる

$$\Rightarrow \text{競合比} \leq \frac{\text{最大の M アイテム} + \text{全 S アイテム}}{\text{最小の M アイテム} + \text{全 S アイテム}} \leq \frac{1/\phi}{1-1/\phi} = \phi = \frac{1+\sqrt{5}}{2}$$

アウトライン

1. オンラインナップサック問題
2. 除去可能設定
3. **資源拡大モデルとバッファモデル**
4. まとめ

資源拡大モデル

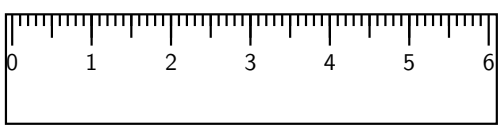
通常の競合比ではアルゴリズムの性能をうまく測れない

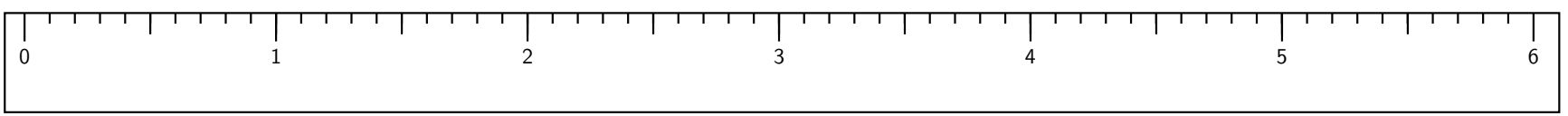
⇨ オンラインアルゴリズムが使える資源を増やして競合比解析

- ページング問題においてキャッシュサイズを拡大 [Sleator and Tarjan 1985]
- k サーバ問題においてサーバ数を増やす [Mannase, McGeoch, and Sleator 1990]
- スケジューリング問題においてマシンの処理速度や台数を増やす [Berman and Coulston 1998; Anand, Garg, and Kumar 2012]
- ビンパッキング問題においてビンの容量を増やす [Azar et al. 2002; Csirik and Woeginger 2002]

ナップサックの容量を R (> 1) に増加 [Iwama and Zhang 2007]

$$\text{競合比} = \frac{\text{容量 } 1 \text{ のナップサックでのオフライン最適値}}{\text{容量 } R \text{ のナップサックでのアルゴリズムの利得}}$$

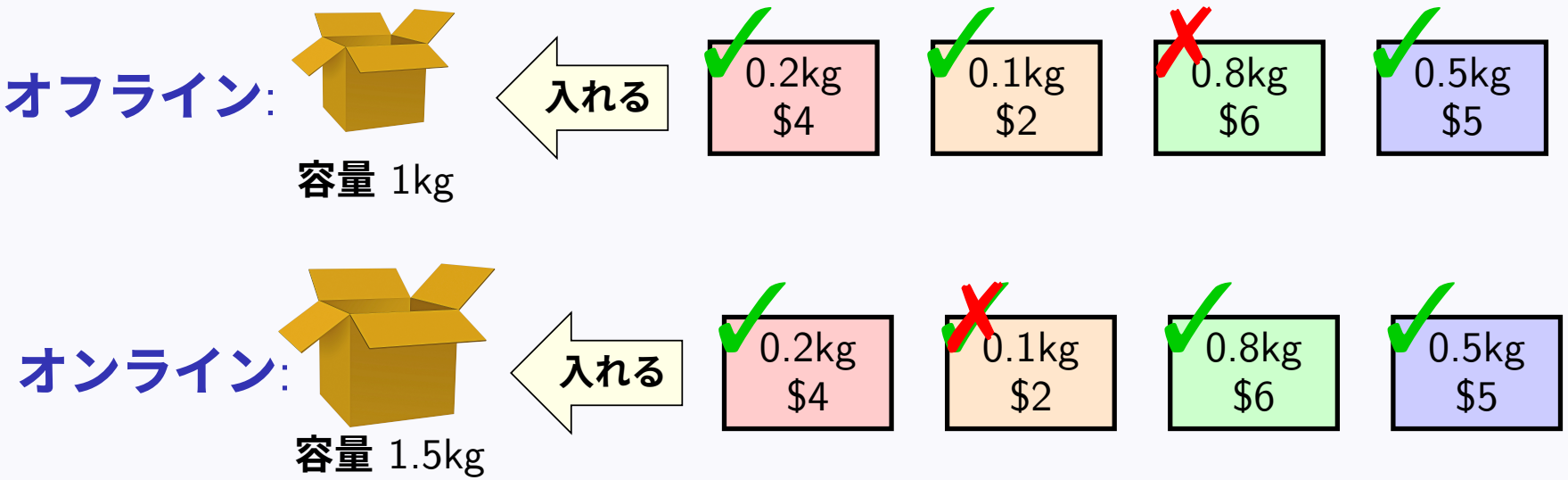




資源拡大の例 (除去可能モデル)

- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のナップサックを使用

例: $R = 1.5$



資源拡大の例 (除去可能モデル)

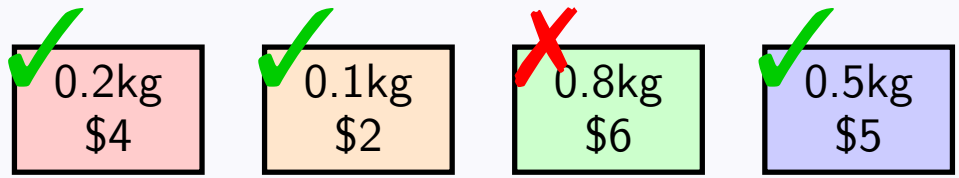
- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のナップサックを使用

例: $R = 1.5$

オフライン:

容量 1kg

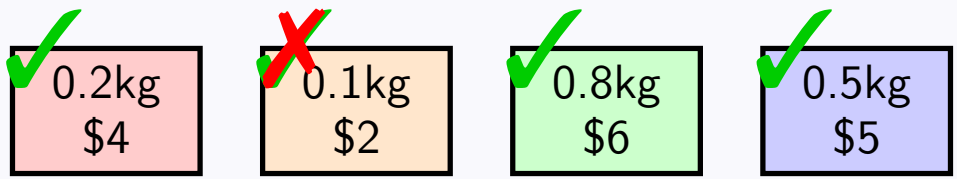
入れる



オンライン:

容量 1.5kg

入れる



競合比 $= \dfrac{4+2+5}{4+6+5} = \dfrac{11}{15}$ (1 **より小さくなることもある**)

資源バッファモデル [Han, Kawase, Makino, Yokomaku 2019]

- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のナップサックを使用
  ただし，入力終了後に容量 1 に収まるようにアイテムを減らす

性質

- オンラインアルゴリズムの方がよい解をとることはない

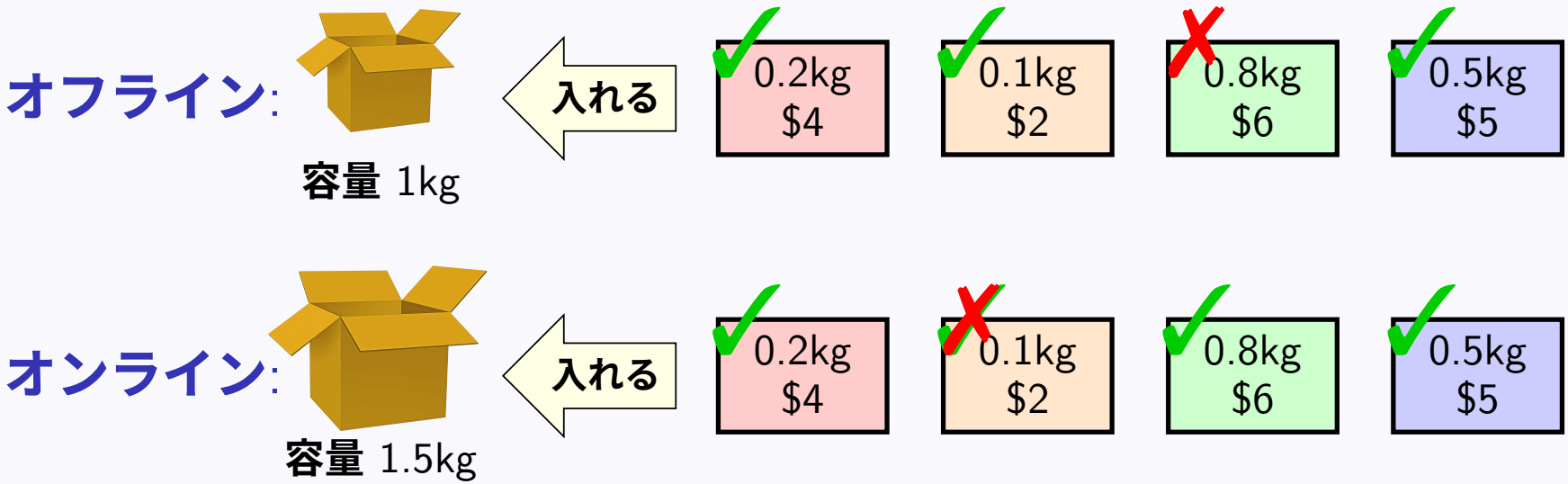
  資源拡大とは異なる
- オフラインアルゴリズムはバッファを利用しても得できない

  バッファ付きの状況に置ける通常の競合比ともみられる

資源バッファの例 (除去可能モデル)

- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のバッファを使用
  ただし，入力終了後に容量 1 に収まるようにナップサックへ詰め替え

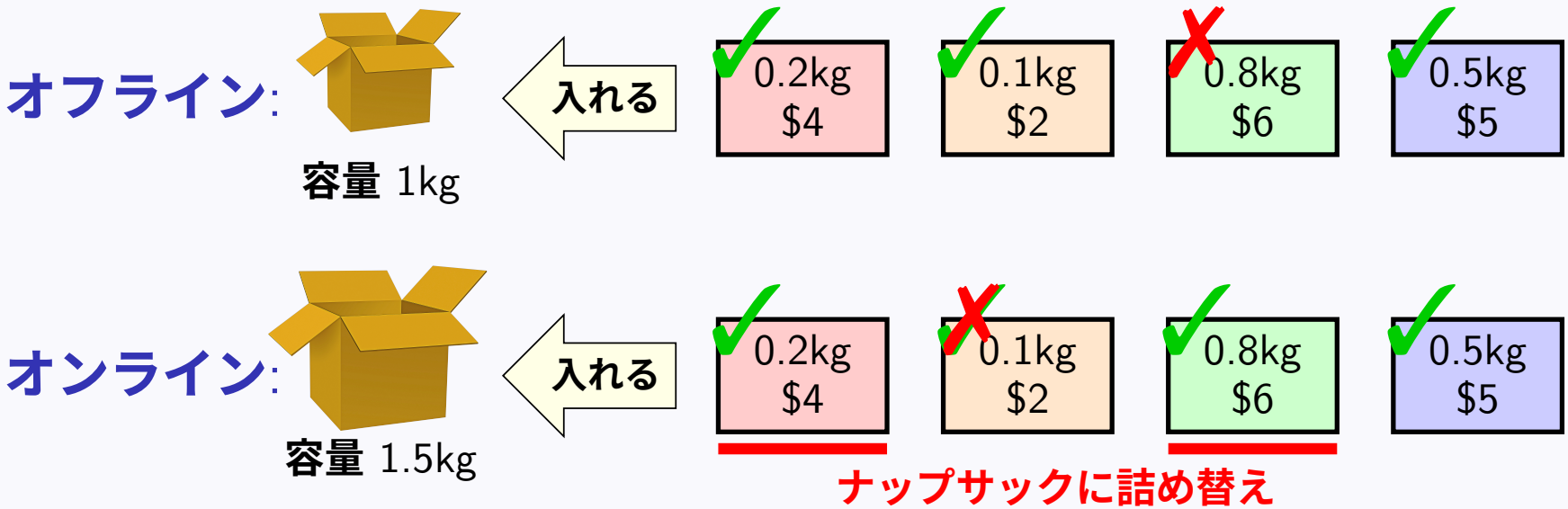
例: $R = 1.5$



資源バッファの例 (除去可能モデル)

- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のバッファを使用
  ただし，入力終了後に容量 1 に収まるようにナップサックへ詰め替え

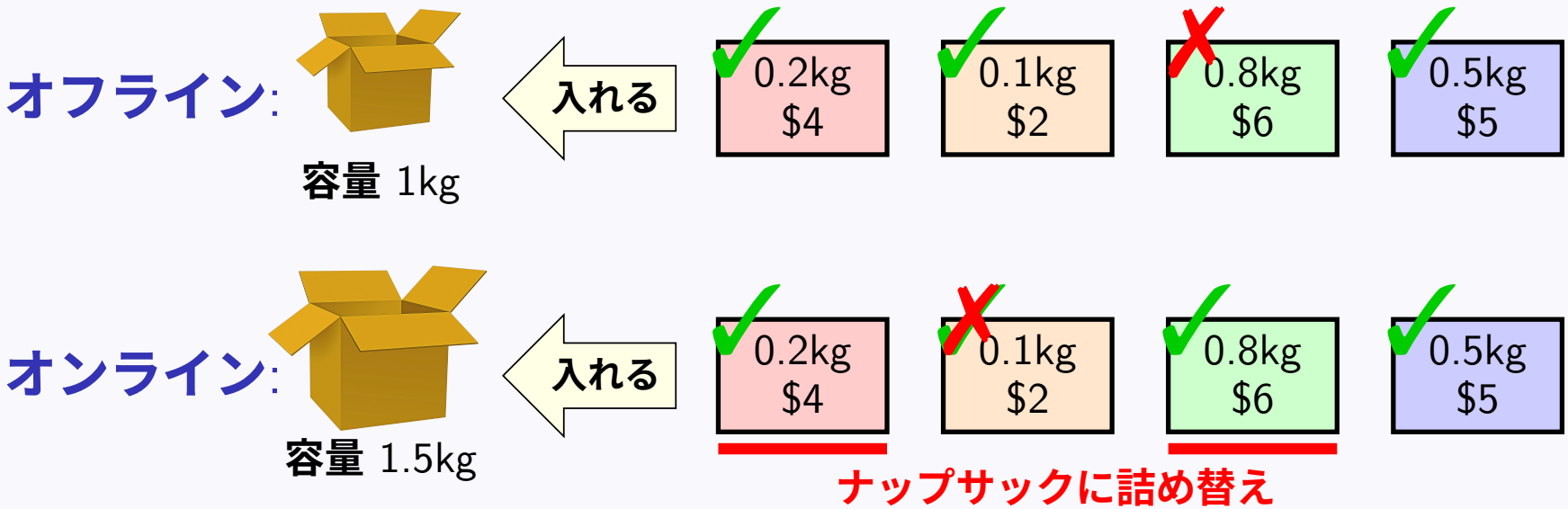
例: $R = 1.5$



資源バッファの例 (除去可能モデル)

- オフラインアルゴリズムは容量 1 のナップサックを使用
- オンラインアルゴリズムは容量 R のバッファを使用
  ただし，入力終了後に容量 1 に収まるようにナップサックへ詰め替え

例: $R = 1.5$

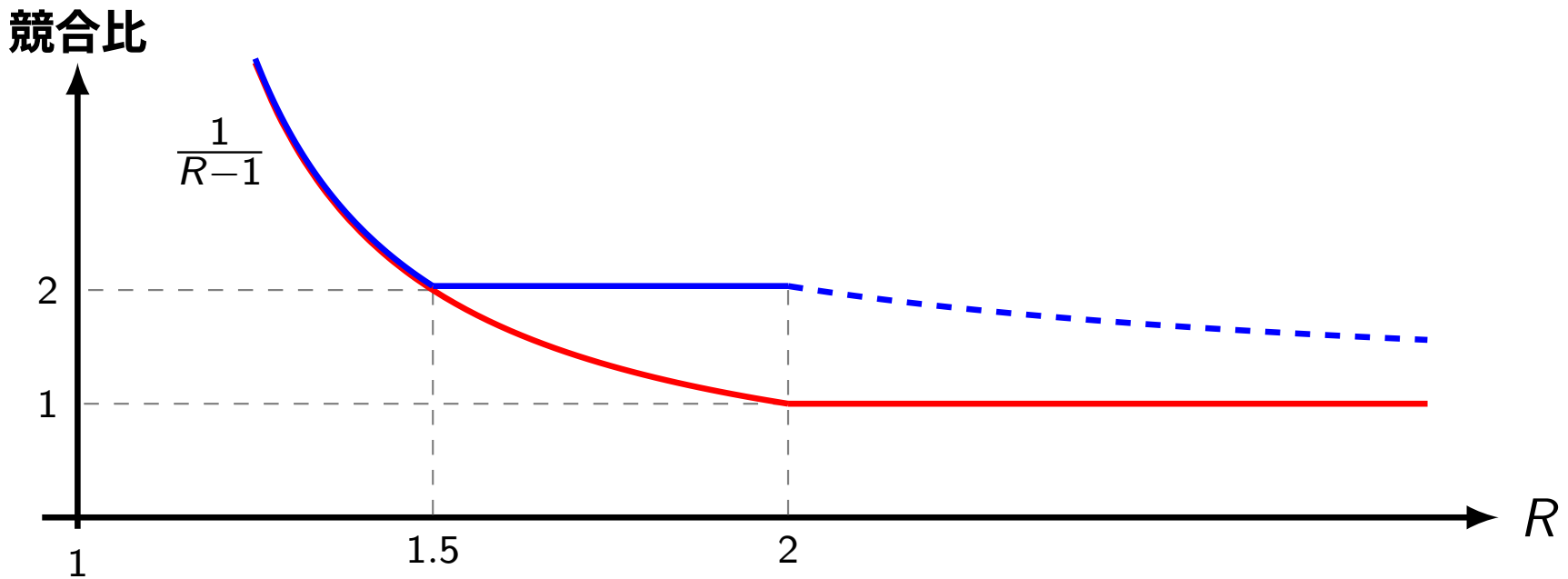


競合比 $= \dfrac{4+2+5}{4+6} = \dfrac{11}{10}$

資源バッファ・資源拡大の結果 (除去可能モデル)

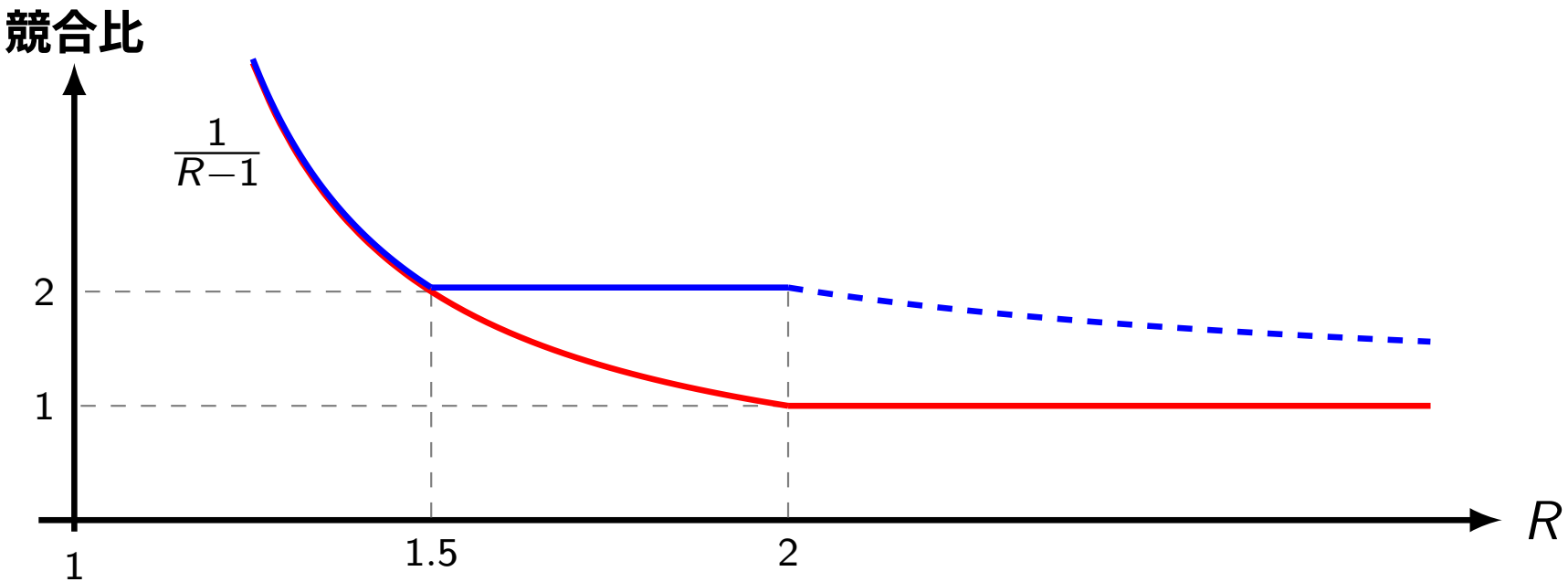
R	資源バッファ	資源拡大
1	∞ [MV1995]	∞ [MV1995]
$(1, 1.5]$	$1/(R-1)$ [HKMY2019]	$1/(R-1)$ [IZ2007]
$[1.5, 2]$	2 [HKMY2019]	$1/(R-1)$ [IZ2007]
$[2, \infty)$	$1+\tilde{\Theta}(1/R)$ [HKMY2019]	1 [IZ2007]

LB: $1+\frac{1}{R+1}$, UB: $1+O(\frac{\log R}{R})$



資源拡大 (除去可能モデル)

R	資源バッファ	資源拡大
1	∞ [MV1995]	∞ [MV1995]
$(1, 1.5]$	$1/(R-1)$ [HKMY2019]	$1/(R-1)$ [IZ2007]
$[1.5, 2]$	2 [HKMY2019]	$1/(R-1)$ [IZ2007]
$[2, \infty)$	$1+\tilde{\Theta}(1/R)$ [HKMY2019]	1 [IZ2007]



資源拡大での競合比 $1/(R-1)$ のアルゴリズム

密度に基づく貪欲アルゴリズム

$B_0 \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $B_i \leftarrow \emptyset$;
  foreach $e \in B_{i-1} \cup \{e_i\}$ (**密度の降順**) **do**
    if **重さ**$(B_i) +$ **重さ**$(e) \leq R$ **then** $B_i \leftarrow B_i \cup \{e\}$;
return B_n;

価値(e)/重さ(e)

資源拡大での競合比 $1/(R-1)$ のアルゴリズム

密度に基づく貪欲アルゴリズム

$B_0 \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $B_i \leftarrow \emptyset$;
  foreach $e \in B_{i-1} \cup \{e_i\}$ (**密度の降順**) **do**
    if **重さ**$(B_i) +$ **重さ**$(e) \leq R$ **then** $B_i \leftarrow B_i \cup \{e\}$;
return B_n;

価値(e)/重さ(e)

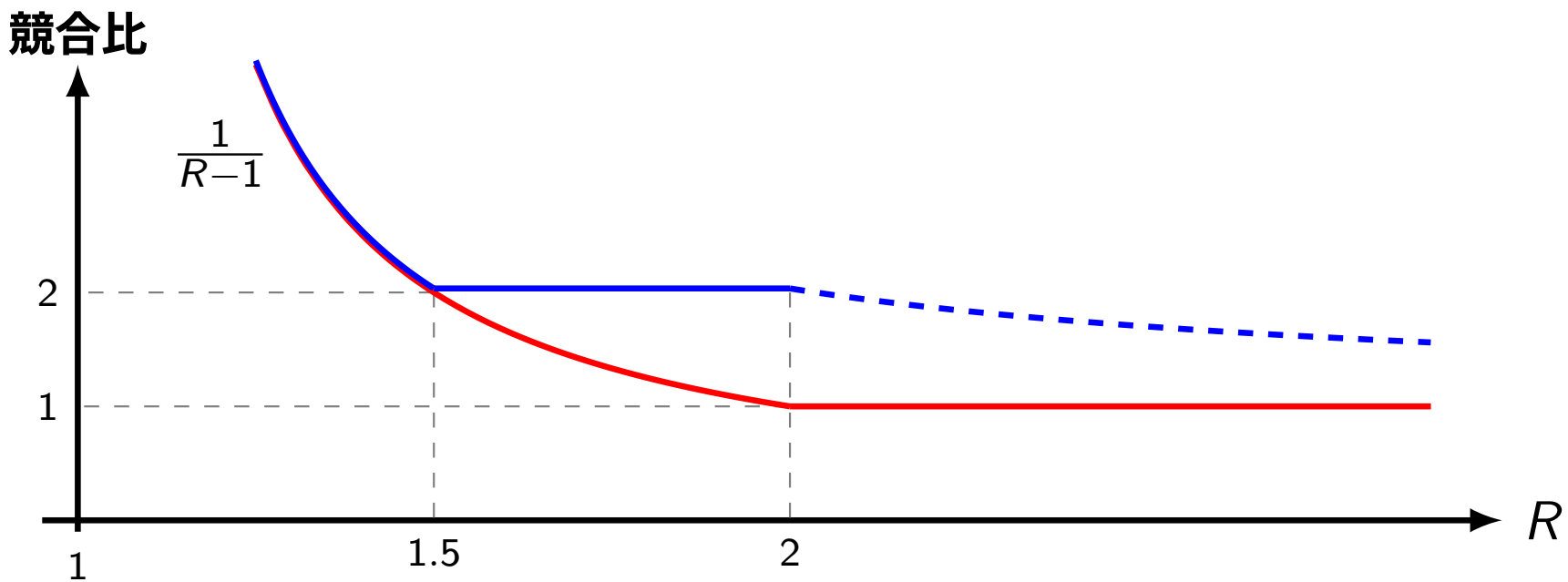
定理 [Iwama and Zhang 2007]

上記のアルゴリズムの競合比は $\max\{1/(R-1), 1\}$ **である**

- **入力アイテムの重さの合計が** R **以下なら競合比** 1
- R **より大きいなら，密度が高いものから重さ** $R-1$ **はとれるので競合比** $1/(R-1)$ **以下**

資源バッファ (除去可能モデル)

R	資源バッファ	資源拡大
1	∞ [MV1995]	∞ [MV1995]
$(1, 1.5]$	$1/(R-1)$ [HKMY2019]	$1/(R-1)$ [IZ2007]
$[1.5, 2]$	2 [HKMY2019]	$1/(R-1)$ [IZ2007]
$[2, \infty)$	$1+\tilde{\Theta}(1/R)$ [HKMY2019]	1 [IZ2007]



除去可能資源バッファ ($R \leq 2$)

密度に基づく貪欲アルゴリズム

$B_0 \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $B_i \leftarrow \emptyset$;
  foreach $e \in B_{i-1} \cup \{e_i\}$ (**密度の降順**) **do**
    if **重さ**$(B_i) + $**重さ**$(e) \leq R$ **then** $B_i \leftarrow B_i \cup \{e\}$;
return $X^* \in \arg\max\{$ **価値**$(X) \mid X \subseteq B_n,$ **重さ**$(X) \leq 1\}$;

価値(e)/**重さ**(e)

除去可能資源バッファ ($R \leq 2$)

密度に基づく貪欲アルゴリズム

$B_0 \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $B_i \leftarrow \emptyset$;
  foreach $e \in B_{i-1} \cup \{e_i\}$ (**密度の降順**) **do**
    if **重さ**$(B_i) +$ **重さ**$(e) \leq R$ **then** $B_i \leftarrow B_i \cup \{e\}$;
return $X^* \in \arg\max\{$ **価値**$(X) \mid X \subseteq B_n,$ **重さ**$(X) \leq 1\}$;

価値(e)/重さ(e)

定理 [Han, Kawase, Makino, and Yokomaku 2019]

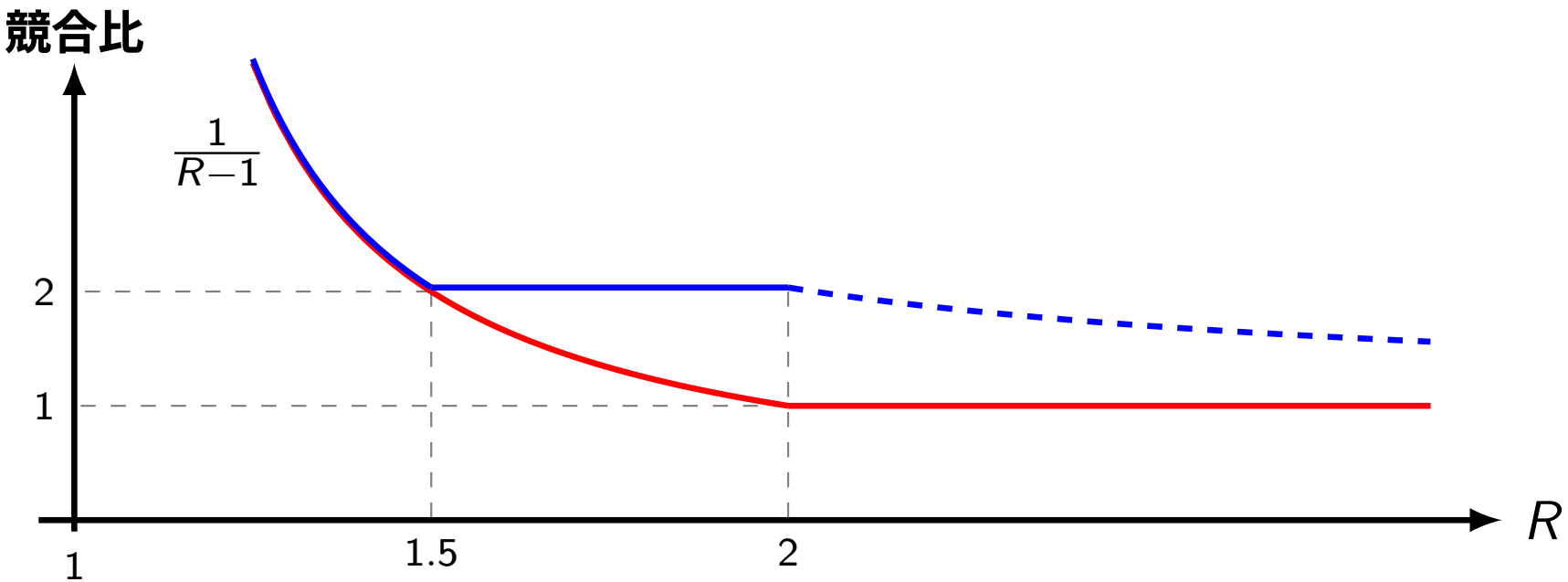
上記のアルゴリズムの競合比は $\max\{1/(R-1), 2\}$ **である**

密度の降順で容量 1 **をはじめて超えるアイテム** e^* **について場合分け**

- $e^* \notin B_n$: $\Rightarrow$ **資源拡大と同様の解析で競合比** $1/(R-1)$
- $e^* \in B_n \Rightarrow$ **オフライン** 2 **近似アルゴリズムと同様の解析で競合比** 2

資源バッファ (除去可能モデル)

R	資源バッファ	資源拡大
1	∞ [MV1995]	∞ [MV1995]
$(1, 1.5]$	$1/(R-1)$ [HKMY2019]	$1/(R-1)$ [IZ2007]
$[1.5, 2]$	2 [HKMY2019]	$1/(R-1)$ [IZ2007]
$[2, \infty)$	$1+\tilde{\Theta}(1/R)$ [HKMY2019]	1 [IZ2007]

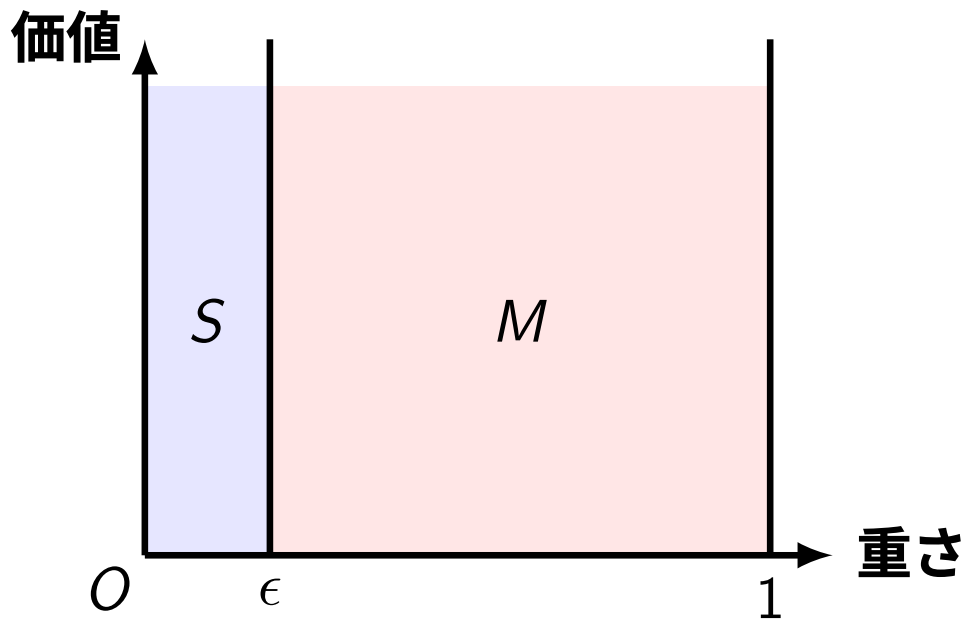


除去可能資源バッファ (一般の R)

アイテムを分割

$O(\log R/R)$

- S: **重さ**$(e) \leq \epsilon$**なるアイテム** e **の集合**
- M: **重さ**$(e) > \epsilon$ **なるアイテム** e **の集合**

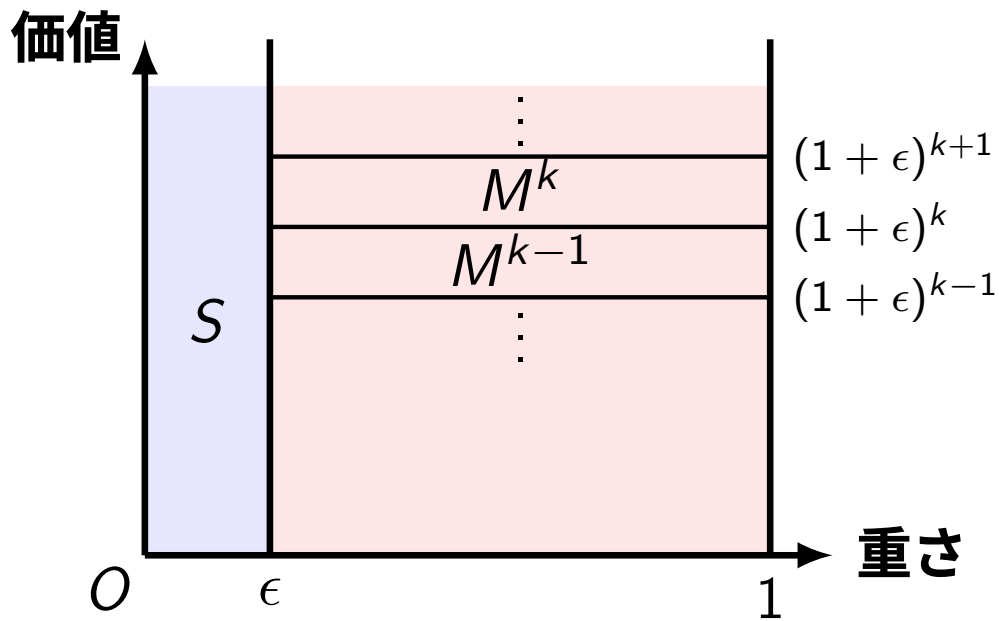


除去可能資源バッファ (一般の R)

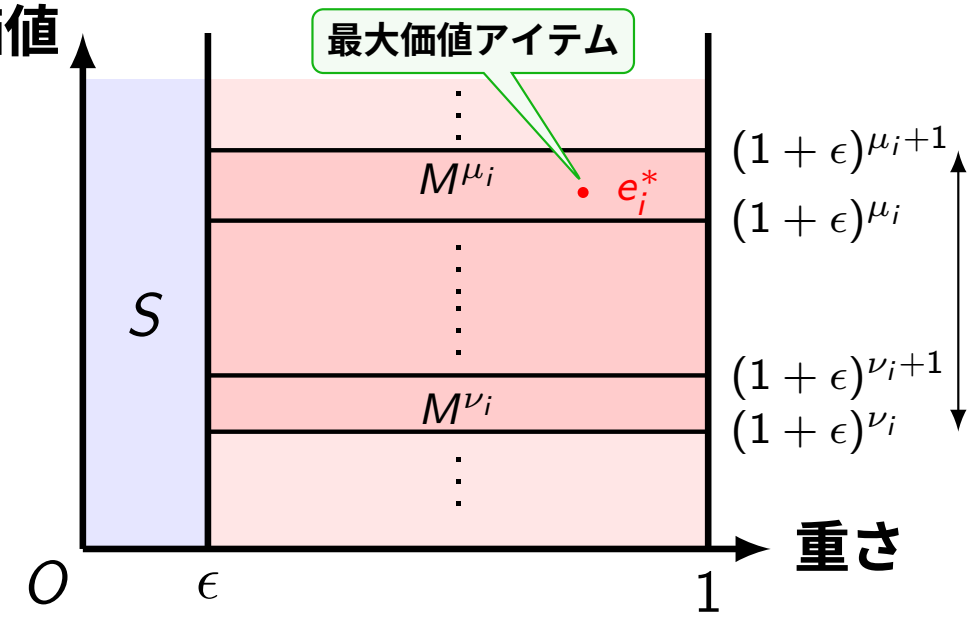
アイテムを分割

$O(\log R/R)$

- S: **重さ**$(e) \leq \epsilon$**なるアイテム** e **の集合**
- M: **重さ**$(e) > \epsilon$ **なるアイテム** e **の集合**
  - M^k: $(1+\epsilon)^k \leq$ **価値**$(e) < (1+\epsilon)^{k+1}$ **なるアイテム** $e \in M$ **の集合**



除去可能資源バッファ (一般の R)

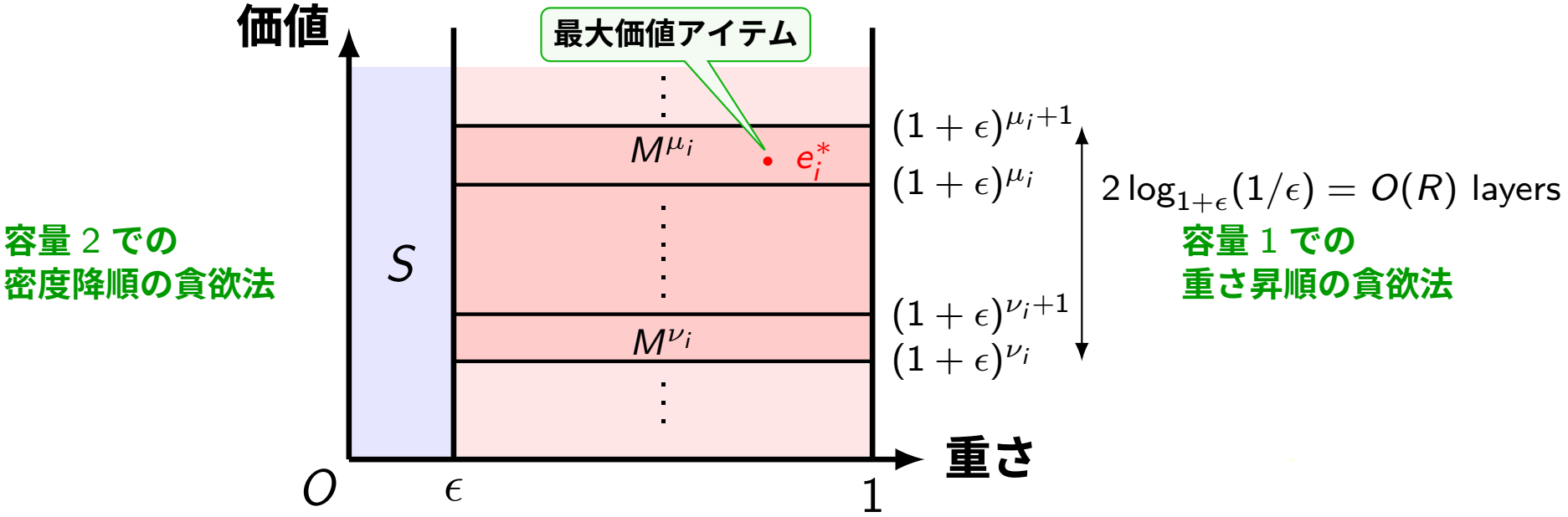


容量 2 での
密度降順の貪欲法

容量 1 での
重さ昇順の貪欲法

$B_0 \leftarrow \emptyset$;
for $i \leftarrow 1, 2, \ldots, n$ **do**
  $B_i \leftarrow \emptyset$ and $B'_i \leftarrow (B_{i-1} \cup \{e_i\})$;
  foreach $e \in B'_i \cap S$ *(***密度の降順***)* **do**
    $B_i \leftarrow B_i \cup \{e\}$;
    if **重さ**$(B_i) > 2$ **then break**;
  Let $e_i^* \in \arg\max\{$ **価値**$(e) \mid e \in B'_i\}$;
  Let $\mu_i \leftarrow \lfloor \log_{1+\epsilon}$ **価値**$(e_i^*) \rfloor$ and $\nu_i \leftarrow \lfloor \log_{1+\epsilon} \epsilon^2 \cdot$ **価値**$(e_i^*) \rfloor$;
  for $j \leftarrow \nu_i, \ldots, \mu_i$ **do**
    foreach $e \in B'_i \cap M^j$ *(***重さの降順***)* **do**
      if **重さ**$(B_i \cap M^j) +$ **重さ**$(e) \leq 1$ **then** $B_i \leftarrow B_i \cup \{e\}$;

除去可能資源バッファ (一般の R)



定理 [Han, Kawase, Makino, and Yokomaku 2019]

上記のアルゴリズムの競合比は $1 + O(\epsilon) = 1 + O(\log R/R)$

略証: **最適解** OPT **に対応するような良い解** $B^* \subseteq B_n$ **をとれる**

- **価値**$(\mathrm{OPT} \cap (\bigcup_{k<\nu_n} M^k)) \leq \epsilon \cdot$ **価値**(OPT)
- **価値**$(\mathrm{OPT} \cap M^k)) \leq (1+\epsilon)$ **価値**$(B^* \cap M^k)$ $(\nu_n \leq \forall k \leq \mu_n)$
- **価値**$(\mathrm{OPT} \cap S) \leq$ **価値**$(B^* \cap S) + \epsilon(1+2\epsilon)$ **価値**(OPT)

アウトライン

1. オンラインナップサック問題
2. 除去可能設定
3. 資源拡大モデルとバッファモデル
4. まとめ

まとめ

- 様々なオンラインナップサック問題のモデルとアルゴリズムを紹介
- ナップサック制約は最も基本的な制約で，多くの問題に現れるため，
  オンラインナップサック問題の解析はオンライン最適化の基盤となる
- 一般的なモデルを標準的な指標で考えるとうまく評価できないため，
  状況に応じた適切なモデルと適切な評価指標を選択する必要がある

まとめ

- 様々なオンラインナップサック問題のモデルとアルゴリズムを紹介
- ナップサック制約は最も基本的な制約で，多くの問題に現れるため，
  オンラインナップサック問題の解析はオンライン最適化の基盤となる
- 一般的なモデルを標準的な指標で考えるとうまく評価できないため，
  状況に応じた適切なモデルと適切な評価指標を選択する必要がある

ご清聴ありがとうございました