数理・計算科学特論 C 第 3 回

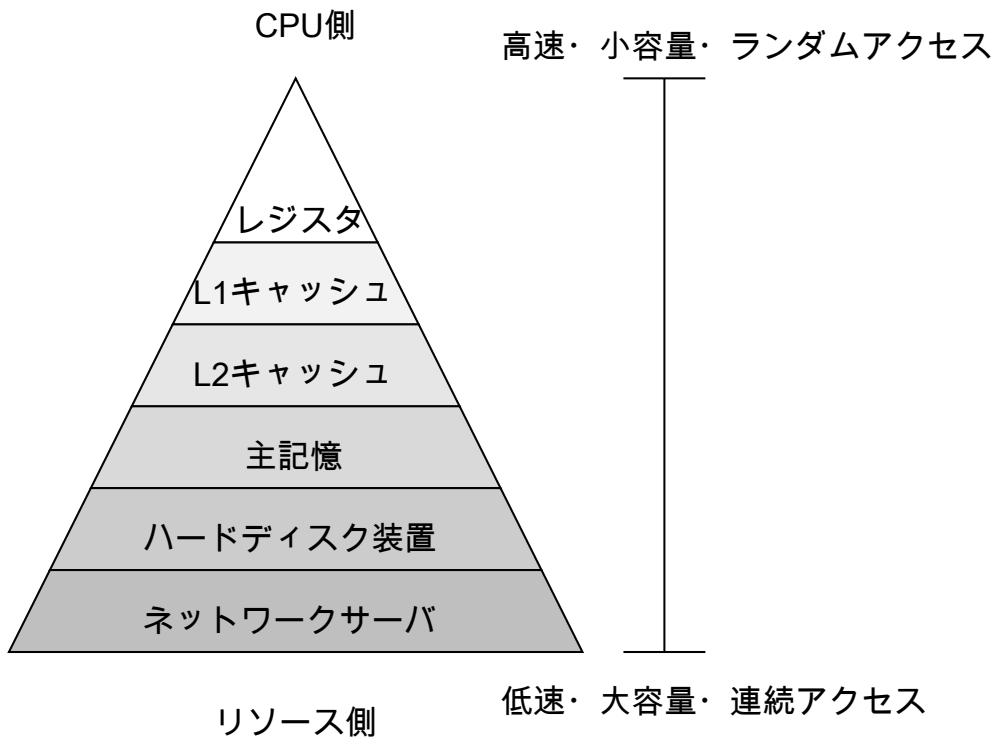
河瀬 康志

2021 年 6 月 15 日

アウトライン

計算機の記憶装置

- 計算機の記憶装置はいくつかの階層に分かれている
- 高速なものほど容量が小さい
- 高速な記憶装置に保管するデータをどのように選べば良いか？



https://ja.m.wikipedia.org/wiki/ファイル:Memory_hierarchy.svg

ページング方式とページング問題

ページング

- 「ページ」と呼ばれる単位でデータをやりとりするメモリ管理のやり方
- ページは全て同じ大きさ

ページング問題

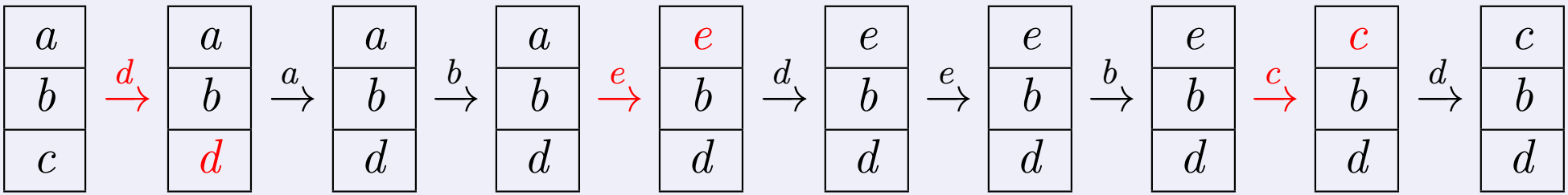
- 記憶装置は 2 段階
  - 低速記憶装置には N 個のページが蓄えられている
  - 高速記憶装置には k 個のページを蓄えることができる $(k < N)$
- 必要データを含むページを要求するページ要求が逐次的に与えられる
- 要求のあったページが高速記憶装置に入っていない場合は，
  低速記憶装置から高速記憶装置にデータをロードする必要がある
  (ページフォルト)
- ページフォルトの数をなるべく小さくしたい

最適オフラインアルゴリズム OPT

ページフォルトが起きた時，将来の要求がもっとも遅いページを削除
（演習：なぜ最適か？）

例 ($k = 3$)

- 高速記憶装置の初期データ：$\{a, b, c\}$
- ページ要求列：$d, a, b, e, d, e, b, c, d$



ページフォルト: 3 回

ページング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 k 以上

証明: データページを $k+1$ 個だけ使い，
ALG が高速記憶装置に保持していないページの要求列を考える

- ALG は毎回ページフォルトを起こす
- OPT（ページフォルトが起きた時，将来のページ要求がもっとも遅いデータを削除）
  - $k+1$ 個しかデータはないので，ページフォルトを起こした時点で考えている $k+1$ 個のページ以外の要求はない
  - 将来のページ要求がもっとも遅いものは k 個以上先

  $\rightsquigarrow$ ページフォルトは k 回に 1 回しか起こさない

$\longrightarrow$ 競合比は k 以上

ページング問題の競合比の下界

定理

任意の(ALG)オンラインアルゴリズムの競合比は k 以上

証明: データページを $k+1$ 個だけ使い，
ALG が高速記憶装置に保持していないページの要求列を考える

- ALG は毎回ページフォルトを起こす
- OPT（ページフォルトが起きた時，将来のページ要求がもっとも遅いデータを削除）
  - $k+1$ 個しかデータはないので，ページフォルトを起こした時点で考えている $k+1$ 個のページ以外の要求はない
  - 将来のページ要求がもっとも遅いものは k 個以上先

  $\rightsquigarrow$ ページフォルトは k 回に 1 回しか起こさない

$\longrightarrow$ 競合比は k 以上

ページング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 k 以上

証明: データページを $k+1$ 個だけ使い，
ALG が高速記憶装置に保持していないページの要求列を考える

- ALG は毎回ページフォルトを起こす
- OPT（ページフォルトが起きた時，将来のページ要求がもっとも遅いデータを削除）
  - $k+1$ 個しかデータはないので，ページフォルトを起こした時点で考えている $k+1$ 個のページ以外の要求はない
  - 将来のページ要求がもっとも遅いものは k 個以上先

  $\rightsquigarrow$ ページフォルトは k 回に 1 回しか起こさない

→ 競合比は k 以上

ページング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 k 以上

証明: データページを $k+1$ 個だけ使い，
ALG が高速記憶装置に保持していないページの要求列を考える

- ALG は毎回ページフォルトを起こす
- OPT（ページフォルトが起きた時，将来のページ要求がもっとも遅いデータを削除）
  - $k+1$ 個しかデータはないので，ページフォルトを起こした時点で考えている $k+1$ 個のページ以外の要求はない
  - 将来のページ要求がもっとも遅いものは k 個以上先

  $\rightsquigarrow$ ページフォルトは k 回に 1 回しか起こさない

→ 競合比は k 以上

ページングのアルゴリズム

ページフォルトが起きた時に削除するデータを次のように選択

- First-In-First-Out (FIFO): もっとも古くロードされたデータ
- Last-In-Last-Out (LIFO): もっとも新しくロードされたデータ
- Least-Recently-Used (LRU): 最近のページ要求がもっとも古いデータ
- Frequency-Count (FC): ページ要求頻度がもっとも少ないデータ

それぞれの競合比はどうなる？

LIFO: もっとも新しくロードされたデータを削除

定理

LIFO の競合比は無限大

悪い入力例 ($k = 3$)

- 高速記憶装置の初期データ：$\{a, b, c\}$
- ページ要求列：$\underbrace{d, c, d, c, d, c, \ldots}_{M回}$

- LIFO のページフォルト: M 回
- 最適な方法でのページフォルト: 1 回（最初に a を削除）

⟶ 競合比 $\geq \frac{M}{1} \xrightarrow{M \to \infty} \infty$　（一般の k でも同様に悪い例を構成できる）

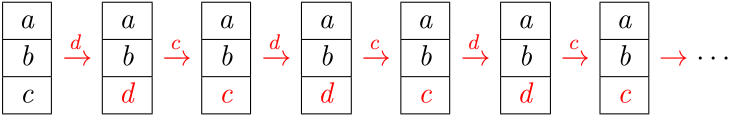
LIFO: もっとも新しくロードされたデータを削除

定理

LIFO の競合比は無限大

悪い入力例 ($k = 3$)

- 高速記憶装置の初期データ：$\{a, b, c\}$
- ページ要求列：$\underbrace{d, c, d, c, d, c, \dots}_{M回}$



- LIFO のページフォルト: M 回
- 最適な方法でのページフォルト: 1 回（最初に a を削除）

→ 競合比 $\geq \frac{M}{1} \xrightarrow{M\to\infty} \infty$　（一般の k でも同様に悪い例を構成できる）

FC: ページ要求頻度がもっとも少ないデータを削除

定理

FC の競合比は無限大

悪い入力例 ($k = 3$)

- 高速記憶装置の初期データ：$\{a, b, c\}$
- ページ要求列：$\underbrace{a, \ldots, a}_{M\text{回}}, \underbrace{b, \ldots, b}_{M\text{回}}, \underbrace{d, c, \ldots, d, c}_{2\times(M-1)\text{回}}$

- FC のページフォルト: $2(M-1)$ 回
- 最適な方法でのページフォルト: 1 回（d が要求された時 a を削除）

→ 競合比 $\geq \frac{2(M-1)}{1} \xrightarrow{M\to\infty} \infty$　（一般の k でも同様に悪い例を構成できる）

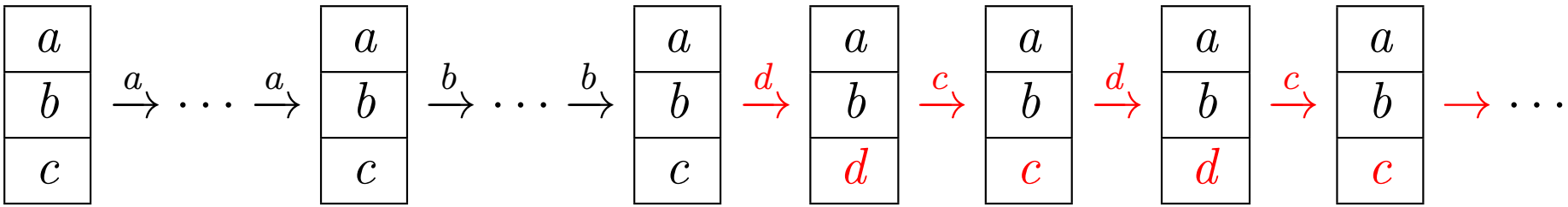
FC: ページ要求頻度がもっとも少ないデータを削除

定理

FC の競合比は無限大

悪い入力例 ($k = 3$)

- 高速記憶装置の初期データ：$\{a, b, c\}$
- ページ要求列：$\underbrace{a, \ldots, a}_{M\text{回}}, \underbrace{b, \ldots, b}_{M\text{回}}, \underbrace{d, c, \ldots, d, c}_{2\times(M-1)\text{回}}$



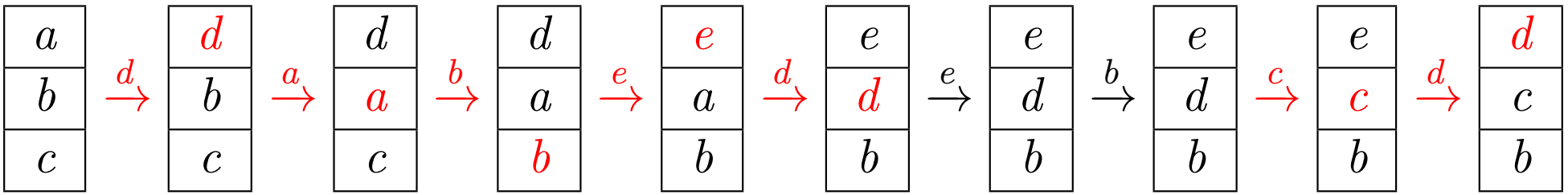
- FC のページフォルト: $2(M-1)$ 回
- 最適な方法でのページフォルト: 1 回（d が要求された時 a を削除）

→ 競合比 $\geq \frac{2(M-1)}{1} \xrightarrow{M\to\infty} \infty$　（一般の k でも同様に悪い例を構成できる）

LRU: 最近のページ要求がもっとも古いデータを削除

例

- 高速記憶装置 $(k = 3)$ の初期データ：$\{a, b, c\}$
- ページ要求列：$d, a, b, e, d, e, b, c, d$



ページフォルト: 7 回

LRU の競合比

定理

LRU の競合比は k

- 下界は k だと知っているので，k 以下であることを示せば良い
- ページ要求列を LRU が k 回ページフォルトを起こす毎 に分けて解析（フェーズ）
- OPT でも各フェーズに 1 回はページフォルトを起こすことを示す

LRU の競合比

定理

LRU の競合比は k

- 下界は k だと知っているので，k 以下であることを示せば良い
- ページ要求列を LRU が k 回ページフォルトを起こす毎 に分けて解析（フェーズ）
- OPT でも各フェーズに 1 回はページフォルトを起こすことを示す

Case 1 フェーズ内で LRU はある p について 2 回ページフォルト

p がロードされてから消去されるまでに k 種類以上のページが要求

2 回ページフォルトするのは，一度目にページフォルトした時にロードしたのに消去した時

→ このフェーズでは $k+1$ 種類以上のページが要求

→ OPT でも 1 回はページフォルトを起こす

LRU の競合比

定理

LRU の競合比は k

- 下界は k だと知っているので，k 以下であることを示せば良い
- ページ要求列を LRU が k 回ページフォルトを起こす毎（フェーズ）に分けて解析
- OPT でも各フェーズに 1 回はページフォルトを起こすことを示す

Case 2 フェーズ内で LRU は k 種類の異なるページについてフォルト

フェーズの開始直前に要求されたページを q とする

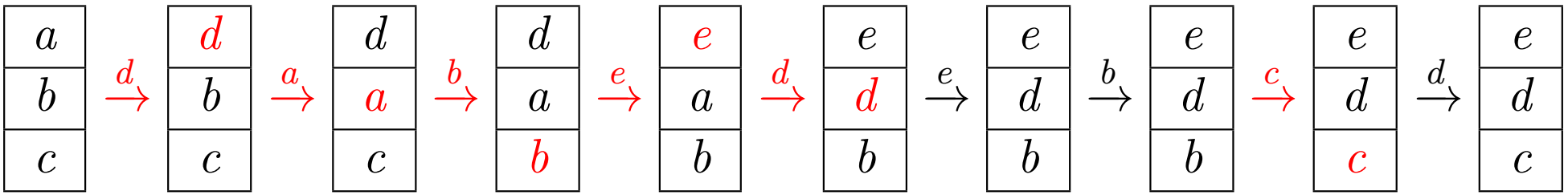
フェーズの開始時には LRU も OPT も q を保持

- LRU が q でフォルトしない
  → q 以外に k 種類の要求があるので OPT はフォルト
- LRU が q でフォルト
  → Case 1 と同様で OPT はフォルト

FIFO: もっとも古くロードされたデータを削除

例

- 高速記憶装置 $(k = 3)$ の初期データ：$\{a, b, c\}$
- ページ要求列：$d, a, b, e, d, e, b, c, d$



ページフォルト: 6 回

定理

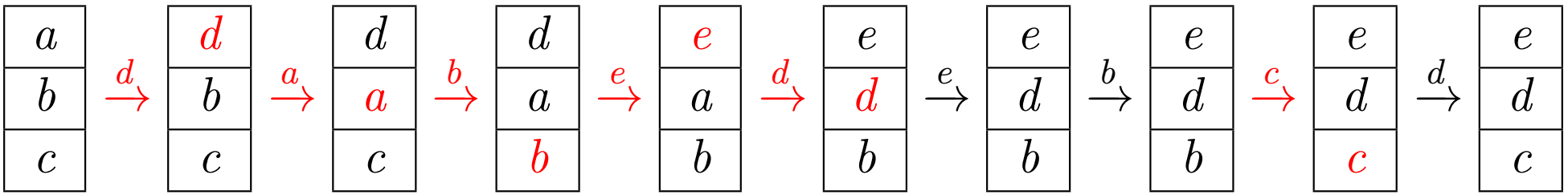
FIFO の競合比は k

証明は LRU と同様 (演習)

FIFO: もっとも古くロードされたデータを削除

例

- 高速記憶装置 $(k = 3)$ の初期データ：$\{a, b, c\}$
- ページ要求列：$d, a, b, e, d, e, b, c, d$



ページフォルト: 6 回

定理

FIFO の競合比は k

証明は LRU と同様 (演習)

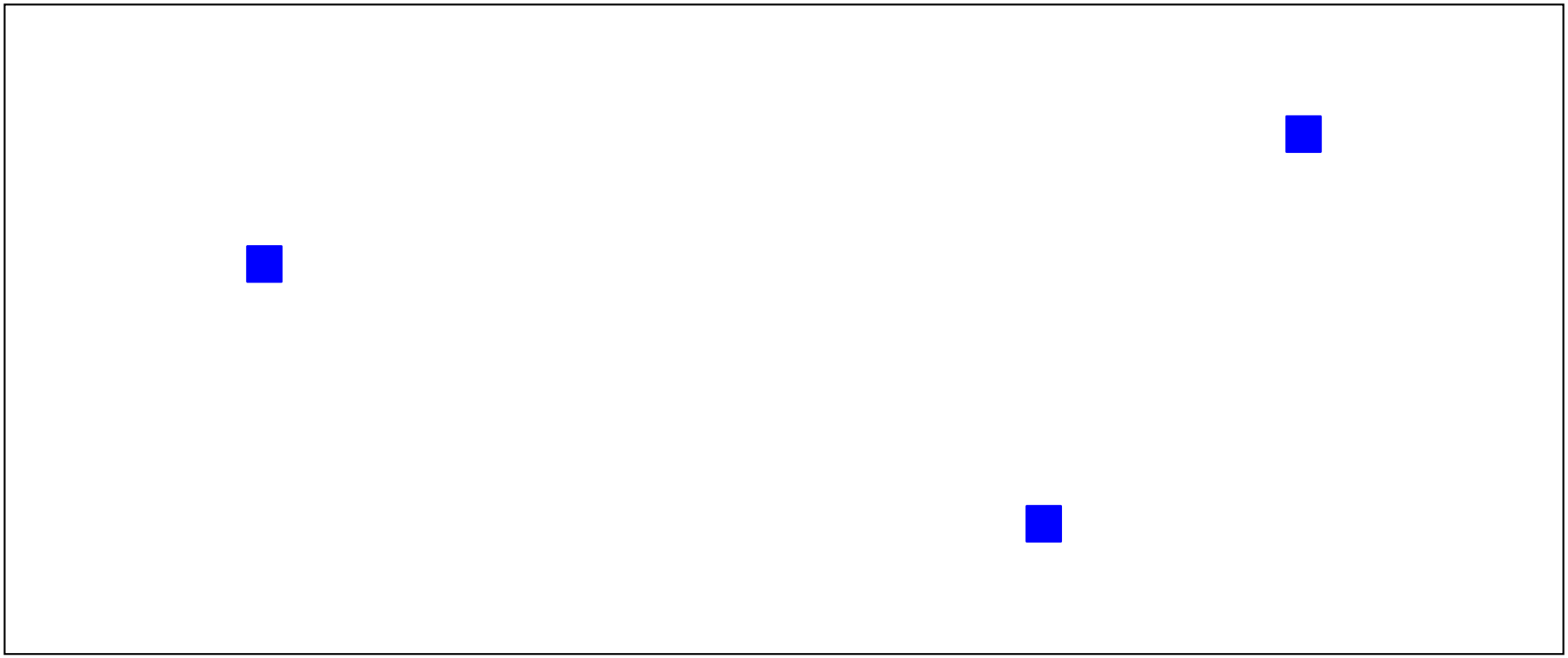
ページング問題まとめ

- 任意のアルゴリズムの競合比は k 以上

- First-In-First-Out (FIFO): もっとも古くロードされたデータ
  → 競合比は k
- Last-In-Last-Out (LIFO): もっとも新しくロードされたデータ
  → 競合比は無限大
- Least-Recently-Used (LRU): 最近のページ要求がもっとも古いデータ
  → 競合比は k
- Frequency-Count (FC): ページ要求頻度がもっとも少ないデータ
  → 競合比は無限大

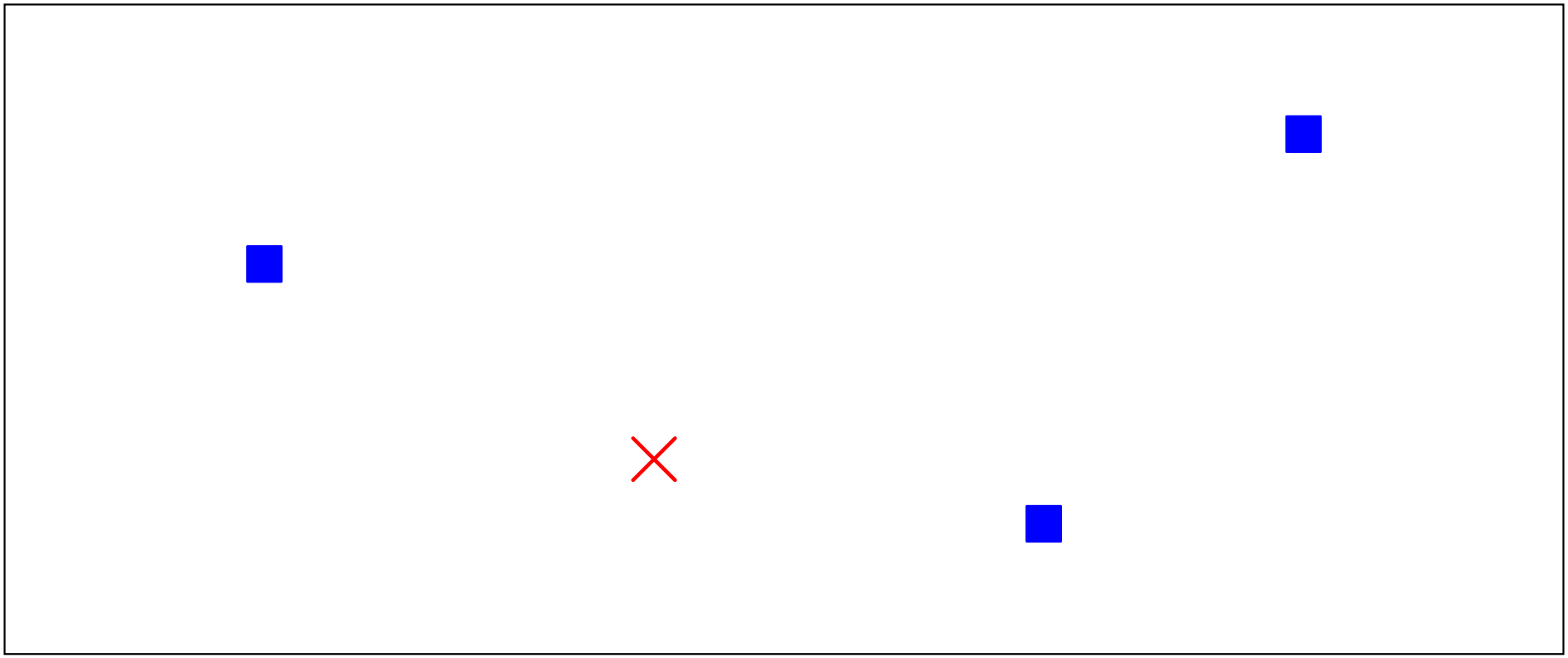
アウトライン

k サーバー問題の例



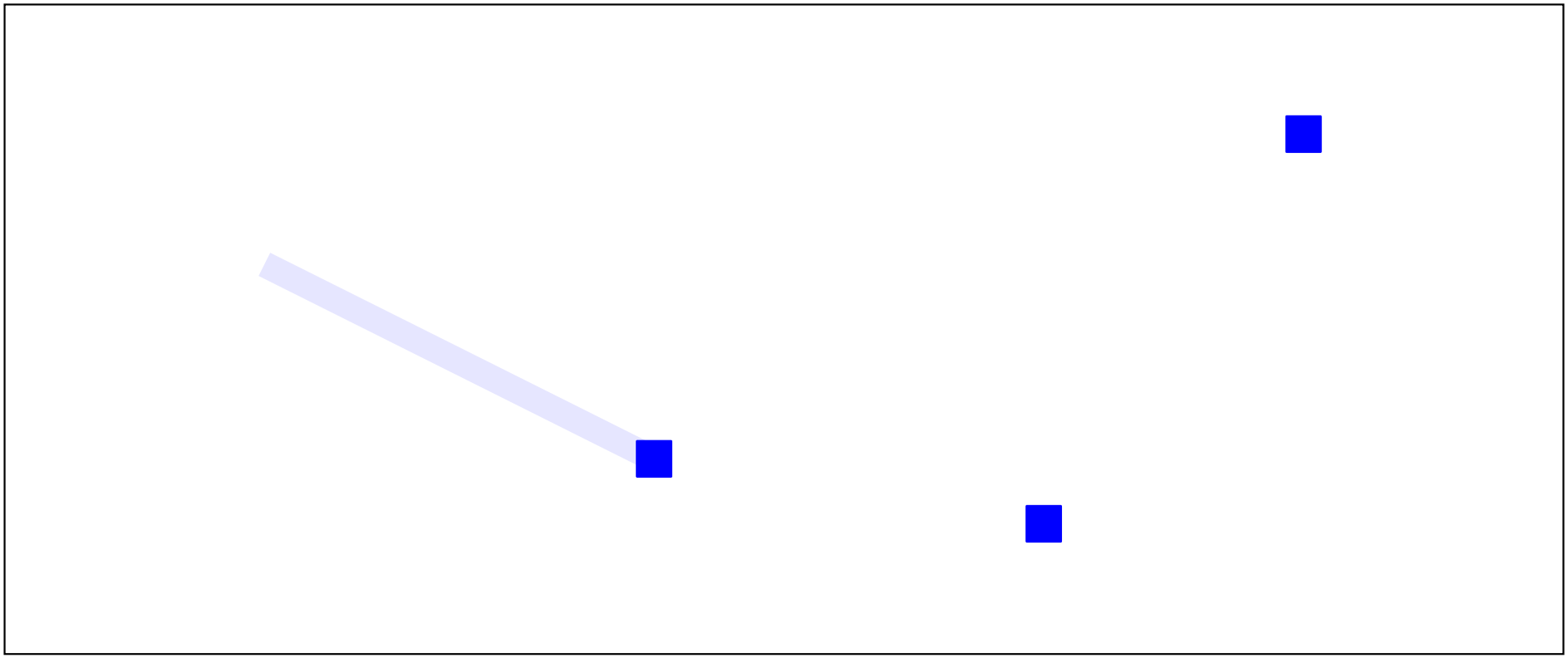
- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



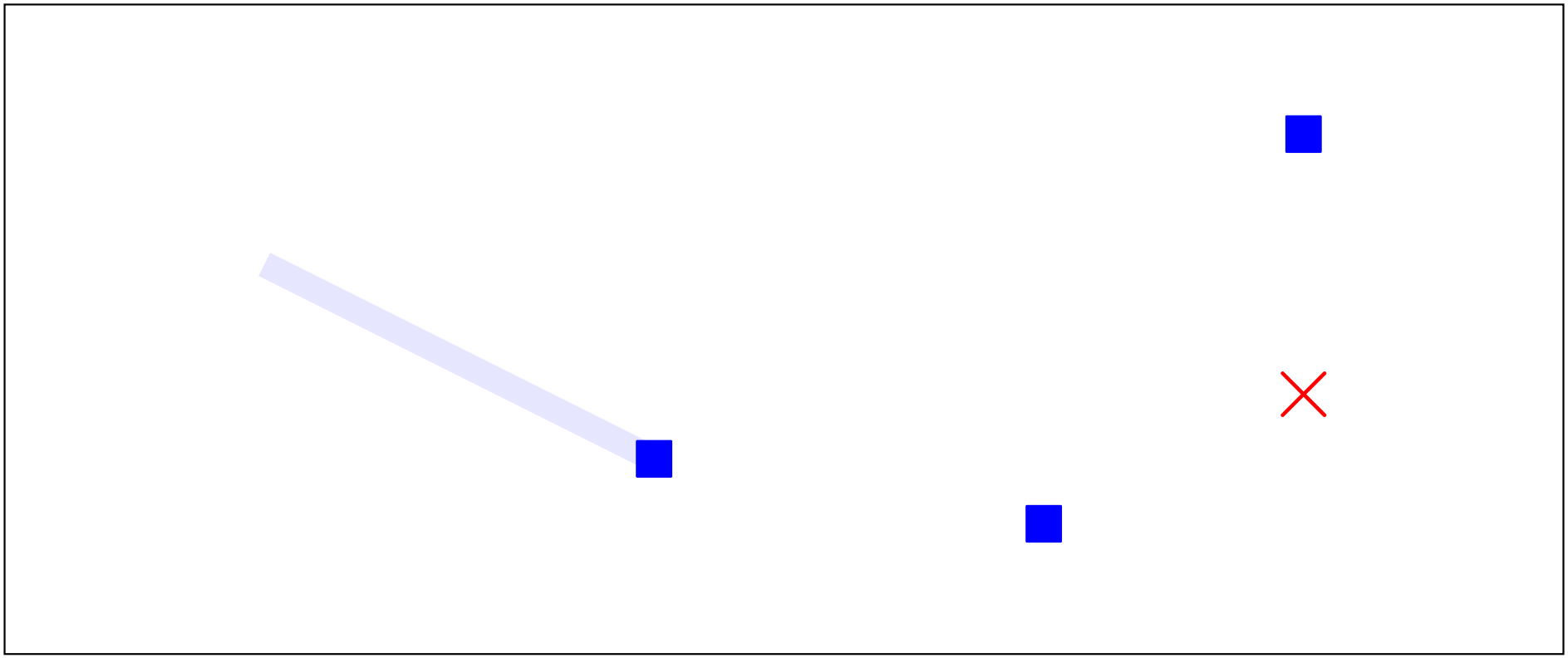
- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



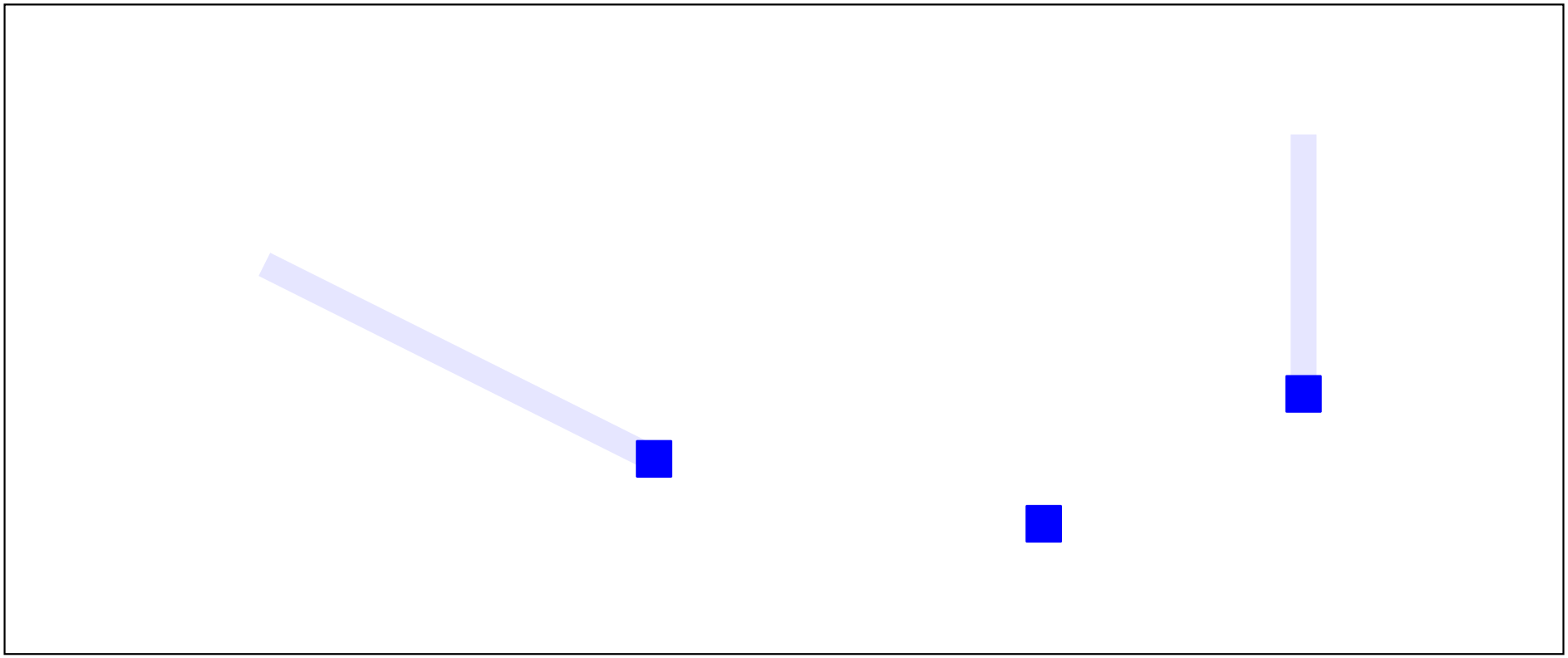
- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



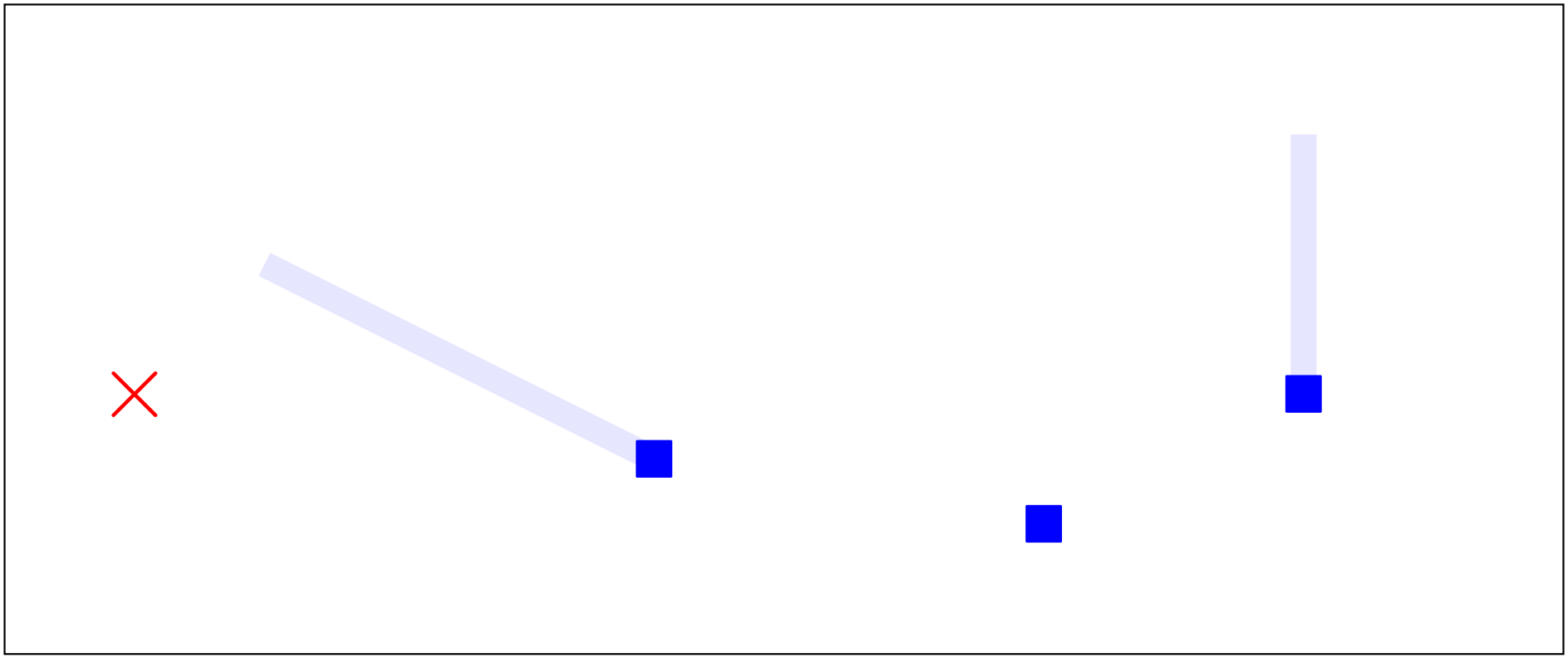
- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



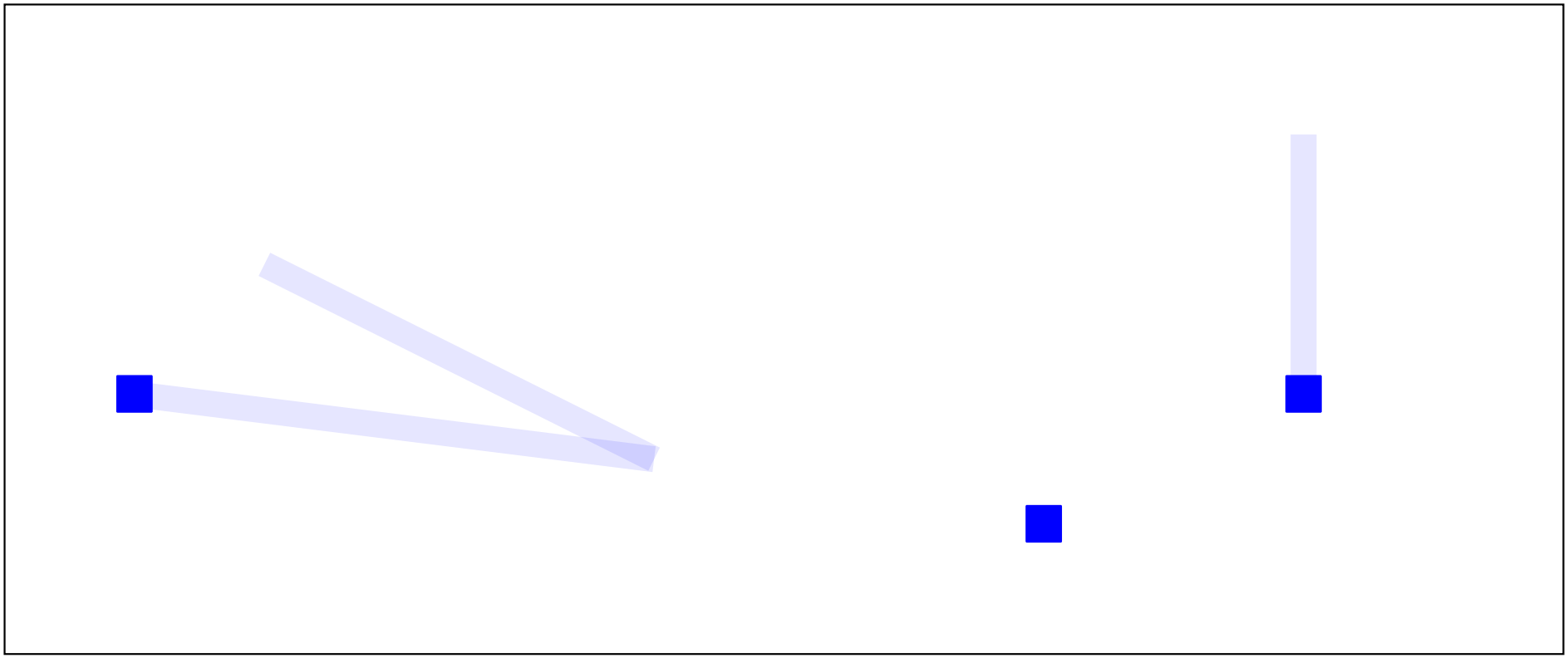
- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の例



- リクエストの発生場所にサーバーを移動する
- 移動距離の合計を最小化したい

k サーバー問題の定義

定義: 距離空間 (P, d)

$d : P \times P \to \mathbb{R}_+$ が次の条件を満たすもの

非退化性 $d(x, y) = 0 \Rightarrow x = y$;

対称性 $d(x, y) = d(y, x)$;

三角不等式 $d(x, y) \leq d(x, z) + d(z, y)$.

k サーバー問題

- 距離空間 (P, d)，k 台のサーバーの初期配置 $s_1, \ldots, s_k \in P$
- 各ステップ t ではリクエスト $r_t \in P$ がやってくる
  → サーバーを 1 台 r_t まで移動させる必要あり
- サーバーの移動距離の合計をなるべく小さくしたい

漸近的競合比

k サーバー問題について漸近的競合比を用いてアルゴリズムを評価する

オンラインアルゴリズム ALG の漸近的競合比

ALG の I に対するコスト

$$\lim_{L\to\infty} \sup_{I:\,\mathrm{OPT}(I)\geq L} \frac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)}$$

I に対する最適コスト

- $\exists\alpha\forall I,\ \mathrm{ALG}(I) \leq \rho\cdot\mathrm{OPT}(I)+\alpha$ ならば漸近的競合比は ρ 以下
  (定数部分を無視した競合比)
- 1 以上であり小さいほど嬉しい
- 以後「漸近的」を省略する

競合比の下界

定理

k サーバー問題の競合比は k 以上

証明:

- ページング問題は k サーバー問題の特殊ケース（どの 2 点間も距離 1）
- ページング問題の競合比は k（漸近的な意味でも）

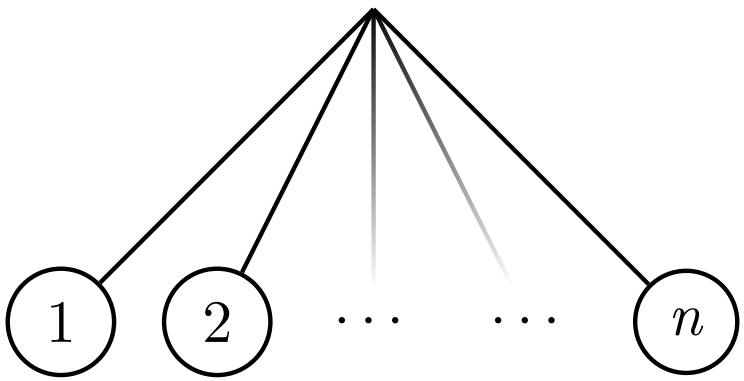
競合比の下界

定理

k サーバー問題の競合比は k 以上

証明:

- ページング問題は k サーバー問題の特殊ケース（どの 2 点間も距離 1）
- ページング問題の競合比は k（漸近的な意味でも）



k サーバー予想

予想 [Manasse, McGeoch, Sleator, 1988]

k サーバー問題の競合比は k

- オンライン問題における重要な未解決問題の 1 つ
- 競合比 $2k-1$ のアルゴリズムは知られている（1994 年〜現在まで最良）
- 距離空間が木であれば，競合比 k のアルゴリズムが知られている

直線上の場合に対する競合比 k のアルゴリズムを紹介する

k サーバー予想

予想 [Manasse, McGeoch, Sleator, 1988]

k サーバー問題の競合比は k

- オンライン問題における重要な未解決問題の 1 つ
- 競合比 $2k-1$ のアルゴリズムは知られている（1994 年〜現在まで最良）
- 距離空間が木であれば，競合比 k のアルゴリズムが知られている

直線上の場合に対する競合比 k のアルゴリズムを紹介する

貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

リクエストにもっとも近いサーバーを移動

インスタンス

- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M \to \infty} \infty$

- 貪欲アルゴリズムのコスト: M

- 最適な方法のコスト: 2

貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

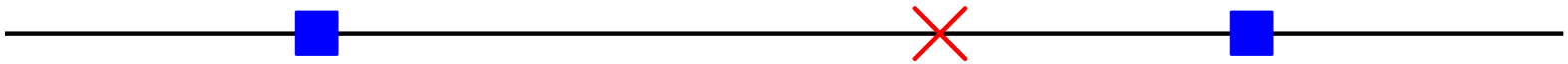
リクエストにもっとも近いサーバーを移動

インスタンス

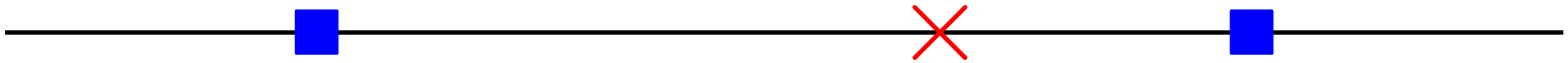
- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M\to\infty} \infty$

- 貪欲アルゴリズムのコスト: M



- 最適な方法のコスト: 2



貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

リクエストにもっとも近いサーバーを移動

インスタンス

- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M \to \infty} \infty$

- 貪欲アルゴリズムのコスト: M

- 最適な方法のコスト: 2

貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

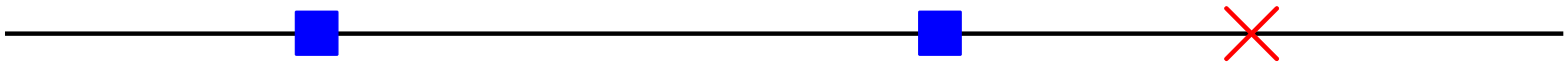
リクエストにもっとも近いサーバーを移動

インスタンス

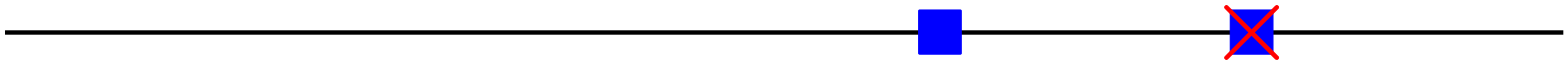
- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M\to\infty} \infty$

- 貪欲アルゴリズムのコスト: M



- 最適な方法のコスト: 2



貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

リクエストにもっとも近いサーバーを移動

インスタンス

- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M\to\infty} \infty$

- 貪欲アルゴリズムのコスト: M

- 最適な方法のコスト: 2

貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

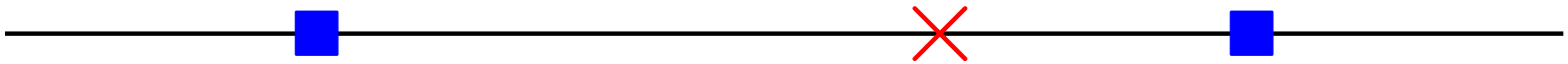
リクエストにもっとも近いサーバーを移動

インスタンス

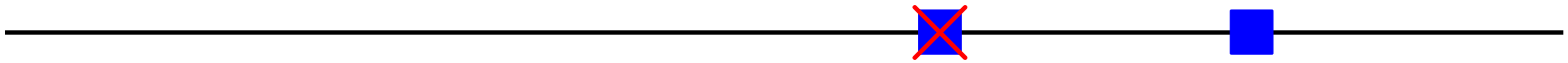
- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M \to \infty} \infty$

- 貪欲アルゴリズムのコスト: M



- 最適な方法のコスト: 2



貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

リクエストにもっとも近いサーバーを移動

インスタンス

- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M \to \infty} \infty$

- 貪欲アルゴリズムのコスト: M

- 最適な方法のコスト: 2

貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

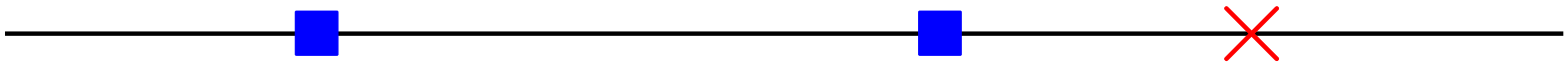
リクエストにもっとも近いサーバーを移動

インスタンス

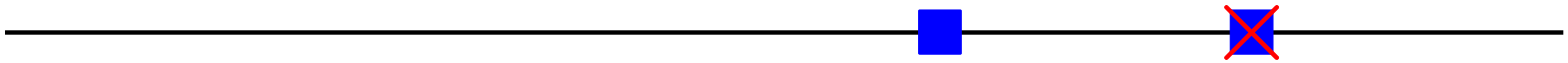
- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

競合比 $\geq M/2 \xrightarrow{M\to\infty} \infty$

- 貪欲アルゴリズムのコスト: M



- 最適な方法のコスト: 2



貪欲アルゴリズム

定理

貪欲アルゴリズムの競合比は，距離空間が直線でも無限大

リクエストにもっとも近いサーバーを移動

インスタンス

- 直線上にサーバーが 2 台（$s_1 = 0,\ s_2 = 3$）
- リクエストの列: $\underbrace{2, 3, 2, 3, \ldots}_{M\text{回}}$

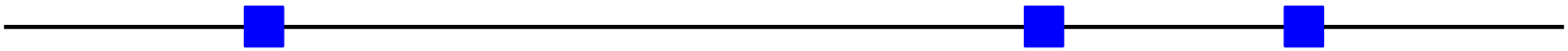
競合比 $\geq M/2 \xrightarrow{M\to\infty} \infty$

- 貪欲アルゴリズムのコスト: M

- 最適な方法のコスト: 2

Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす



DC アルゴリズムの競合比が k であることを示す

Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす

DC アルゴリズムの競合比が k であることを示す

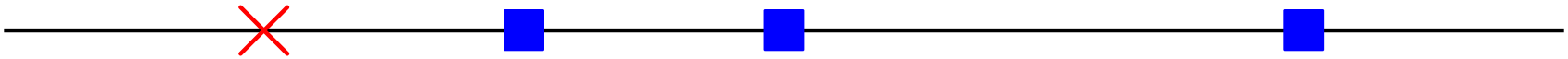
Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす

DC アルゴリズムの競合比が k であることを示す

Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす



DC アルゴリズムの競合比が k であることを示す

Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす

DC アルゴリズムの競合比が k であることを示す

Double Coverage アルゴリズム

- リクエストが 2 つのサーバーの間に来た時
  → 2 つとも同じ距離だけ近づける
- リクエストが k 個のサーバーの外側に来た時
  → 一番近いサーバーだけを動かす

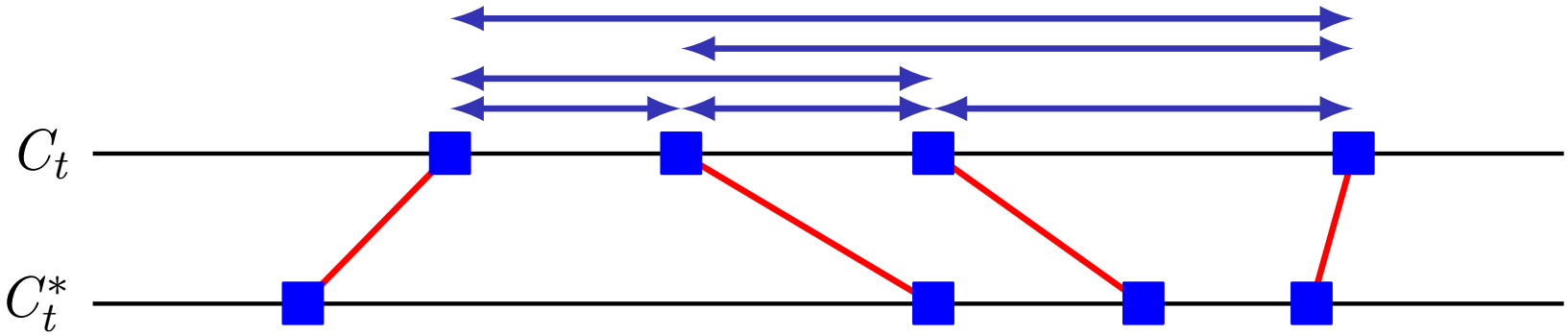
DC アルゴリズムの競合比が k であることを示す

ポテンシャル関数

最小重み完全マッチング　　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

- C_t: t 番目までのリクエスト処理後の DC での配置
- C_t^*: t 番目のリクエスト処理後の OPT での配置



証明の流れ

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

補題 1: C^* を動かした時のポテンシャルの増加量 (後で示す)

$$\Phi(C_{t-1}, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) \leq k \cdot M(C_{t-1}^*, C_t^*)$$

補題 2: C を動かした時のポテンシャルの減少量 (後で示す)

$$\Phi(C_t, C_t^*) - \Phi(C_{t-1}, C_t^*) \leq -M(C_{t-1}, C_t)$$

- 補題を合わせると

OPT の t での移動距離　DC の t での移動距離

$$\Phi(C_t, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) \leq k \cdot M(C_{t-1}^*, C_t^*) - M(C_{t-1}, C_t)$$

- t について和をとると

$$\text{定数} = \Phi(C_T, C_T^*) - \Phi(C_0, C_0^*) \leq k \cdot \text{OPT のコスト} - \text{DC のコスト}$$

→ 競合比 k 以下！

証明の流れ

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

補題 1: C^* を動かした時のポテンシャルの増加量 (後で示す)

$$\Phi(C_{t-1}, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) \leq k \cdot M(C_{t-1}^*, C_t^*)$$

補題 2: C を動かした時のポテンシャルの減少量 (後で示す)

$$\Phi(C_t, C_t^*) - \Phi(C_{t-1}, C_t^*) \leq -M(C_{t-1}, C_t)$$

- 補題を合わせると

OPT の t での移動距離　DC の t での移動距離

$$\Phi(C_t, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) \leq k \cdot M(C_{t-1}^*, C_t^*) - M(C_{t-1}, C_t)$$

- t について和をとると

$$\text{定数} = \Phi(C_T, C_T^*) - \Phi(C_0, C_0^*) \leq k \cdot \text{OPT のコスト} - \text{DC のコスト}$$

→ 競合比 k 以下！

補題 1 の証明

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

補題 1: C^* を動かした時のポテンシャルの増加量

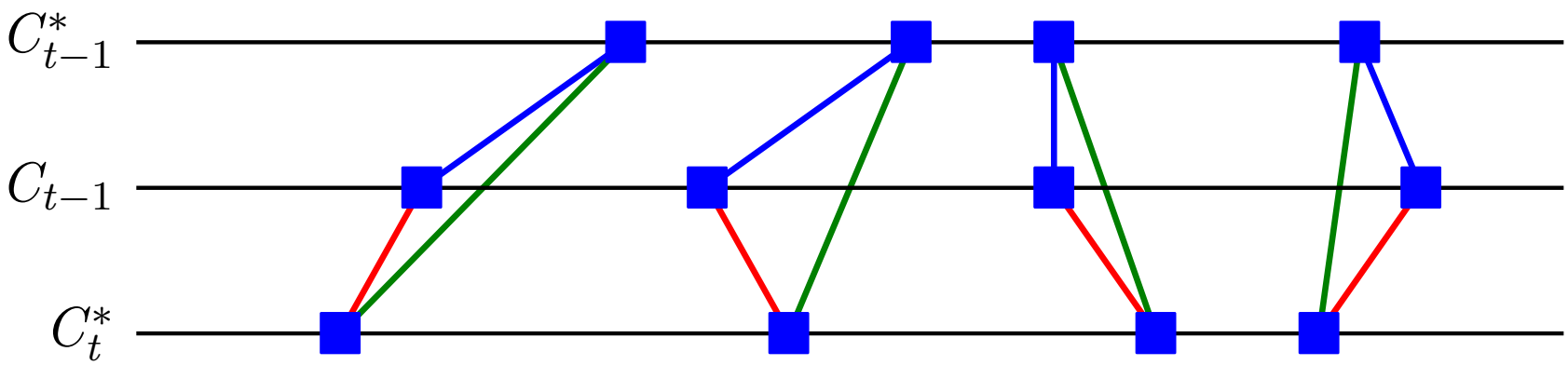
$$\Phi(C_{t-1}, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) \leq k \cdot M(C_{t-1}^*, C_t^*)$$

ポテンシャルの定義

$$\Phi(C_{t-1}, C_t^*) - \Phi(C_{t-1}, C_{t-1}^*) = k \cdot \Big(M(C_{t-1}, C_t^*) - M(C_{t-1}, C_{t-1}^*) \Big)$$

三角不等式

$$\leq k \cdot M(C_{t-1}^*, C_t^*)$$



補題 2 の証明の流れ

最小重み完全マッチング　　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

補題 2: C を動かした時のポテンシャルの減少量

$$\Phi(C_t, C_t^*) - \Phi(C_{t-1}, C_t^*) \leq -M(C_{t-1}, C_t)$$

- C_t と C_{t-1} の間で動いたサーバーが 1 台か 2 台かで場合分け
  （リクエストの位置が k 台の外側か，2 台のサーバーの間か）
- $M(C_t, C_t^*)$ の変化量と $D(C_t)$ の変化量それぞれについて評価する

$M(C_t, C_t^*)$ の変化量（動いたサーバーが 1 台）

最小重み完全マッチング　　サーバーペアの距離和

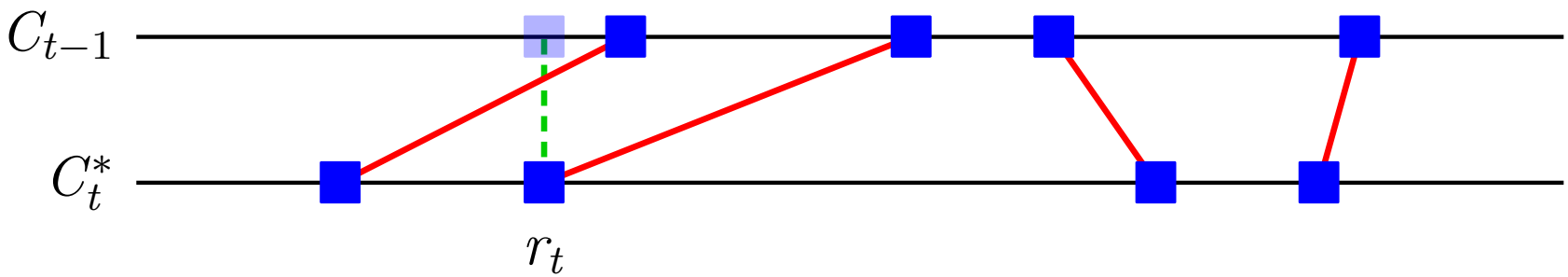
$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

$M(C_t, C_t^*)$ の変化量

動いたサーバーが 1 台の時，$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$

移動距離だけマッチングの距離を減らす

C_t^* では r_t にサーバーが存在することに注意



$D(C_t)$ の変化量（動いたサーバーが 1 台）

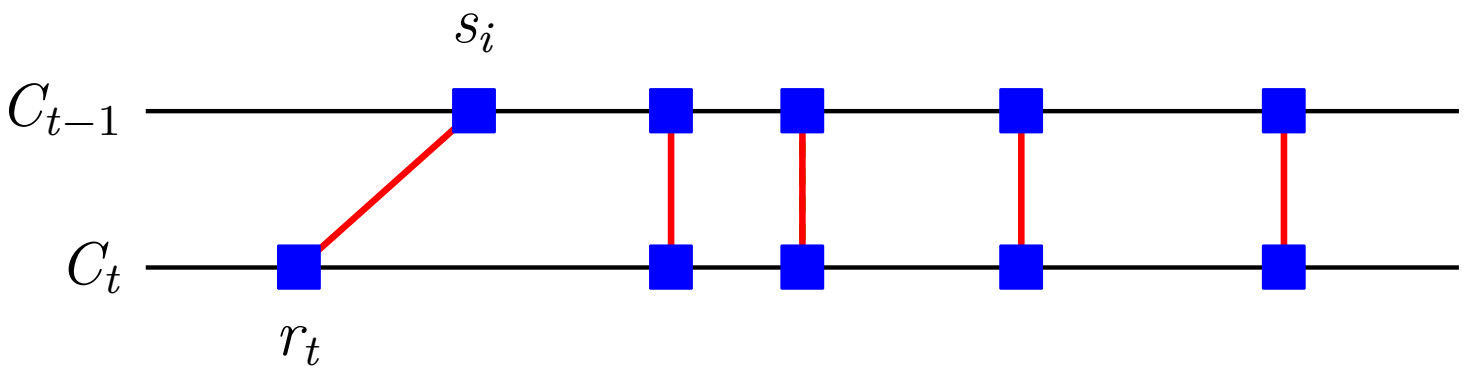
最小重み完全マッチング　　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

$D(C_t)$ の変化量

動いたサーバーが 1 台の時 $D(C_t) = D(C_{t-1}) + (k-1) \cdot M(C_{t-1}, C_t)$

増加量は $(k-1) \cdot d(s_i, r_t) = (k-1) \cdot M(C_{t-1}, C_t)$



補題 2 の証明（動いたサーバーが 1 台）

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

まとめると，動いたサーバーが 1 台の場合，以下が成立

$M(C_t, C_t^*)$ の変化量

$$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

$D(C_t)$ の変化量

$$D(C_t) = D(C_{t-1}) + (k-1) \cdot M(C_{t-1}, C_t)$$

よって，この場合，補題 2 が成立　$M(C_t, C_t^*)$ の変化量の k 倍と $D(C_t)$ の変化量を足せば良い

補題 2

$$\Phi(C_t, C_t^*) \leq \Phi(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

補題 2 の証明（動いたサーバーが 1 台）

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

まとめると，動いたサーバーが 1 台の場合，以下が成立

$M(C_t, C_t^*)$ の変化量

$$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

$D(C_t)$ の変化量

$$D(C_t) = D(C_{t-1}) + (k-1) \cdot M(C_{t-1}, C_t)$$

よって，この場合，補題 2 が成立 $M(C_t, C_t^*)$ の変化量の k 倍と $D(C_t)$ の変化量を足せば良い

補題 2

$$\Phi(C_t, C_t^*) \leq \Phi(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

$M(C_t, C_t^*)$ の変化量（動いたサーバーが 2 台）

最小重み完全マッチング　　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

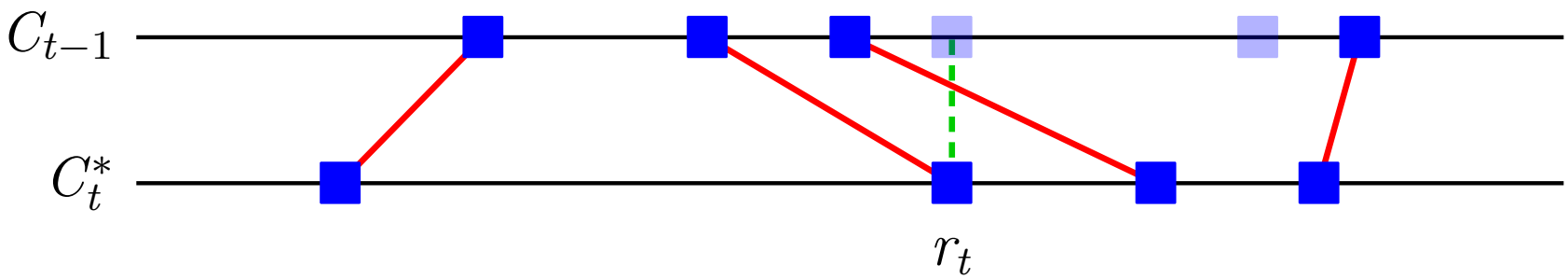
$M(C_t, C_t^*)$ の変化量

動いたサーバーが 2 台の時，$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*)$

少なくとも一方のサーバーは移動距離だけマッチングの距離を減らす

もう一方のサーバーはマッチングの距離を増やしたとしても移動距離だけ

→ マッチングの総距離は増えない



$D(C_t)$ の変化量（動いたサーバーが 2 台）

最小重み完全マッチング　　サーバーペアの距離和

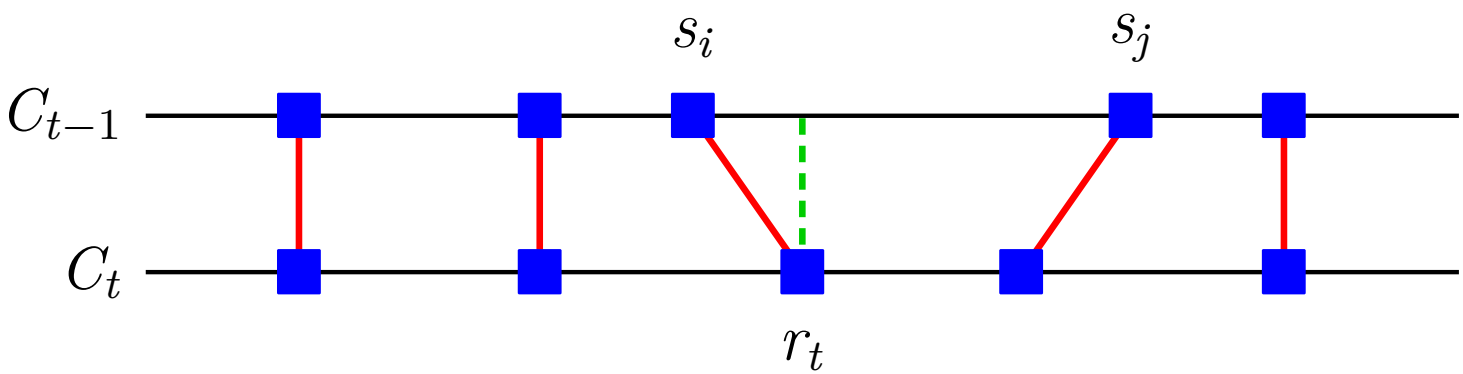
$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

$D(C_t)$ の変化量

動いたサーバーが 2 台の時，$D(C_t) = D(C_{t-1}) - M(C_{t-1}, C_t)$

動いたサーバーのペア以外については，増減がキャンセルアウト

→ 減少量は $2d(s_i, r_t) = M(C_{t-1}, C_t)$



補題 2 の証明（動いたサーバーが 2 台）

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

まとめると，動いたサーバーが 2 台の場合以下が成立

$M(C_t, C_t^*)$ の変化量

$$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*)$$

$D(C_t)$ の変化量

$$D(C_t) = D(C_{t-1}) - M(C_{t-1}, C_t)$$

よって，この場合も，補題 2 が成立　$M(C_t, C_t^*)$ の変化量の k 倍と $D(C_t)$ の変化量を足せば良い

補題 2

$$\Phi(C_t, C_t^*) \leq \Phi(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

補題 2 の証明（動いたサーバーが 2 台）

最小重み完全マッチング　サーバーペアの距離和

$$\Phi(C_t, C_t^*) := k \cdot M(C_t, C_t^*) + D(C_t)$$

まとめると，動いたサーバーが 2 台の場合以下が成立

$M(C_t, C_t^*)$ の変化量

$$M(C_t, C_t^*) \leq M(C_{t-1}, C_t^*)$$

$D(C_t)$ の変化量

$$D(C_t) = D(C_{t-1}) - M(C_{t-1}, C_t)$$

よって，この場合も，補題 2 が成立 $M(C_t, C_t^*)$ の変化量の k 倍と $D(C_t)$ の変化量を足せば良い

補題 2

$$\Phi(C_t, C_t^*) \leq \Phi(C_{t-1}, C_t^*) - M(C_{t-1}, C_t)$$

DC アルゴリズムまとめ

直線上の k サーバー問題に対し

- DC アルゴリズムの競合比は k 以下（ポテンシャルに基づく解析）
- 任意のアルゴリズムの競合比は k 以上（証明略）

定理

直線上の k サーバー問題に対し，DC アルゴリズムの競合比は k

- 木上の場合も同様のアルゴリズムで競合比 k を示すことができる

  リクエスト点との間に他のサーバーがないサーバーを等速度でリクエスト点に向けて動かす

- 一般の距離空間に拡張することは難しい

  t 番目のリクエストが来た時には，「全てのリクエストを処理したのち配置 X にする最適コスト」と「現在の配置 C_{t-1} から配置 X にするコスト」の和を最小化する配置 X に移動することで競合比 $2k-1$ を達成可能 [Koutsoupias, Papadimitriou, 1994]

DC アルゴリズムまとめ

直線上の k サーバー問題に対し

- DC アルゴリズムの競合比は k 以下（ポテンシャルに基づく解析）
- 任意のアルゴリズムの競合比は k 以上（証明略）

定理

直線上の k サーバー問題に対し，DC アルゴリズムの競合比は k

- 木上の場合も同様のアルゴリズムで競合比 k を示すことができる

  リクエスト点との間に他のサーバーがないサーバーを等速度でリクエスト点に向けて動かす

- 一般の距離空間に拡張することは難しい

  t 番目のリクエストが来た時には，「全てのリクエストを処理したのち配置 X にする最適コスト」と「現在の配置 C_{t-1} から配置 X にするコスト」の和を最小化する配置 X に移動することで競合比 $2k-1$ を達成可能 [Koutsoupias, Papadimitriou, 1994]

アウトライン

まとめ

まとめ

- ページング問題

  FIFO, LIFO, LRU, FC の競合比解析

- k サーバー問題

  直線上の場合に対する DC アルゴリズム

演習

- ページング問題に対し「ページフォルトが起きた時，将来の要求がもっとも遅いページを削除」が最適オフラインアルゴリズムとなるのはなぜか？
- ページング問題に対し FIFO の競合比が k となることを示せ．