

# アルゴリズム入門(9) (オンラインアルゴリズム)

宮崎修一  
京都大学 学術情報メディアセンター

# オンラインアルゴリズム

これまでの問題は、入力が全て分かっているが、解の個数が多いのでしらみつぶしに探すのが難しかった。

例えNP完全問題でも、時間さえ掛ければ最適解が求まる。



## オンラインアルゴリズム

入力の一部しか分からないことにより、計算が難しい。時刻とともに入力が次々に与えられる状況で、将来の入力を知らずにその場の判断をしなければならない。

- ・株の売買
- ・新幹線の座席予約
- ・コンピュータ上でのページング
- ・京都市バスの1日乗車券

- ・
- ・
- ・

オンラインアルゴリズムの性能は、**競合比**で表される。

入力 $\sigma$ を最後まで読んだ後なら、最適解を計算できる  
→ $\text{OPT}(\sigma)$

オンラインアルゴリズムAが入力 $\sigma$ に対して得るコスト  
→ $A(\sigma)$

「Aの競合比が $c$ 」とは、全ての $\sigma$ に対して、

$$\frac{A(\sigma)}{\text{OPT}(\sigma)} \leq c$$

(最小化問題の場合)

$$\frac{\text{OPT}(\sigma)}{A(\sigma)} \leq c$$

(最大化問題の場合)

が成り立つこと。つまり、入力を最後まで見た場合の $c$ 倍以内に常に収めることができること。

## 例: スキーレンタル問題

これからスキーを始めるが、今後何回スキーに行くか分からない。  
スキー板を買うべきか、レンタルすべきか？

- ・買うと5万円。
- ・レンタルすると、1回につき1万円。
- ・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)。

### 戦略1: ずっとレンタルし続ける。

100回行ったとき、100万円。

最初に買っていれば、5万円で済んだ。

$100\text{万}/5\text{万} = 20$ 倍悪い。

(つまり競合比は20以上。実際は無限大。)

### 戦略2: 2回目で買う。

2回しか行かなかったとき、 $1+5=6$ 万円。

レンタルしていれば、2万円で済んだ。

$6\text{万}/2\text{万} = 3$ 倍悪い。

(つまり競合比は3以上。)(ちょうど3であることも言える。4)

戦略3: 1~4回の間はレンタル。5回目に行くときに買う。

行く回数が4回以内のとき

行く回数が  $i$  回だったとき

使うのは  $i$  万円

最適解も  $i$  万円 (買うより安い)

→ 全く損をしない。

行く回数が5回以上のとき

使うのは  $4+5=9$  万円

最適は、最初買って5万円

→ 比は  $9/5=1.8$

戦略3の競合比は1.8

もっと良いアルゴリズムはあるか？

戦略3が最適アルゴリズムであることを示す。

アルゴリズムとしては、一度買ったならそれ以降それを使うのがまし。  
なので、「何回目に買うか」でアルゴリズムは決まる。

$k$ 回目に買うとする。

すると、 $k$ 回しか行かないという場合、

かかったお金は $(k-1)+5=k+4$ 万円

・ $1 \leq k \leq 4$ のとき、最適解は、毎回レンタルして $k$ 万円

$$(k+4)/k = 1 + 4/k \geq 2$$

なので、比が1.8よりも悪くなる。

・ $k \geq 6$ のとき、最適解は、最初買って5万円。

$$(k+4)/5 \geq 2$$

なので、比が1.8よりも悪くなる。

よって $k=5$ が最適。

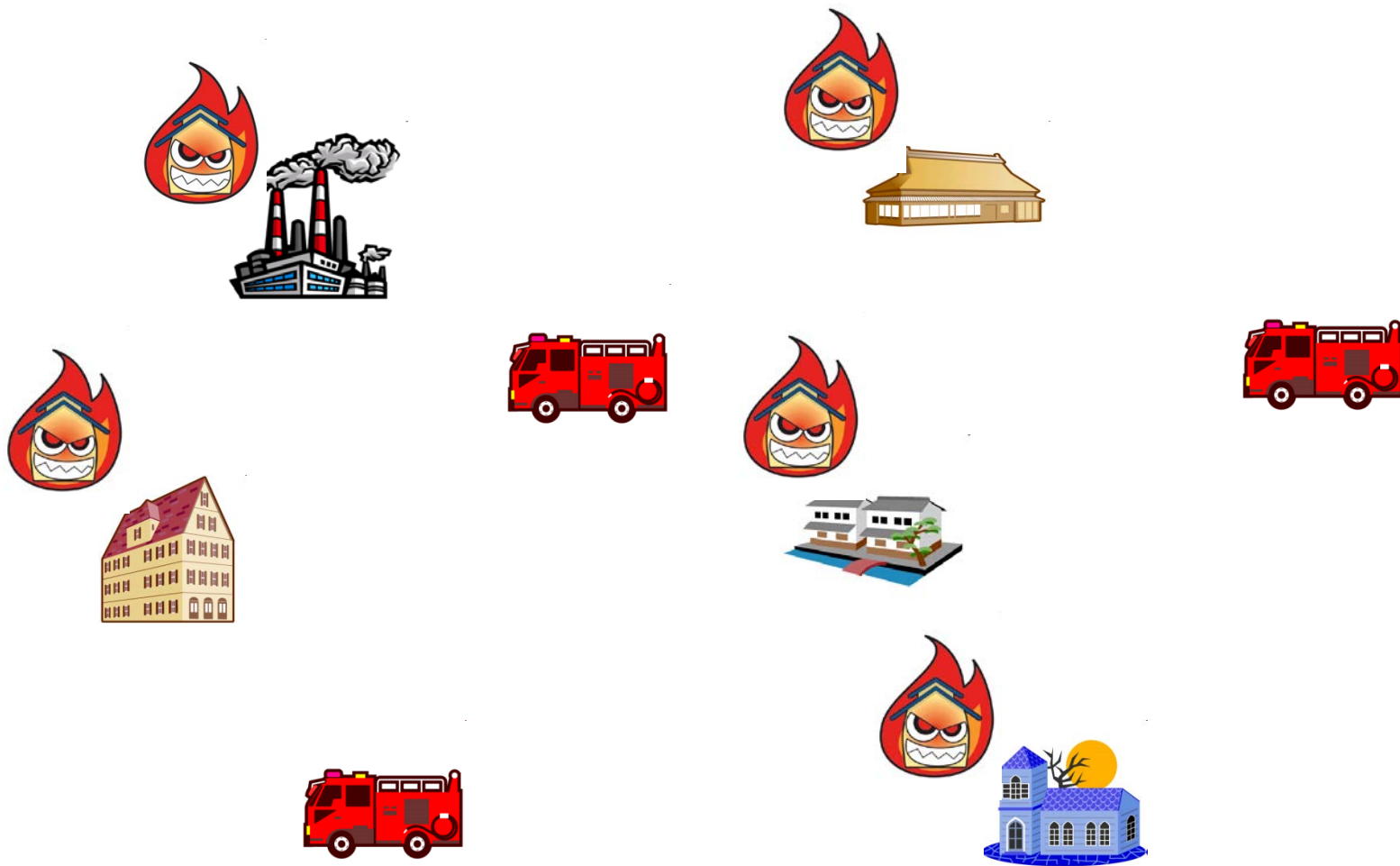
## $k$ サーバ問題

消防車が $k$ 台ある。

火事の起こった場所に1台行かなければならない。

どこで火事が起こるか分からない。(同じ場所で複数回起こることもありうる。)

消防車の総移動距離を最小化する。



直感的に良さそうなもの。

最も近い消防車を行かせる。→ 貪欲アルゴリズム





**問題:** 貪欲アルゴリズムにとって都合の悪い入力はあるか？

最適解は

## 有名な予想 ( $k$ サーバ予想)

「 $k$ サーバ問題の競合比は $k$ 。」

現段階で知られていること。

$k$ サーバ問題の競合比は $2k-1$ 以下。

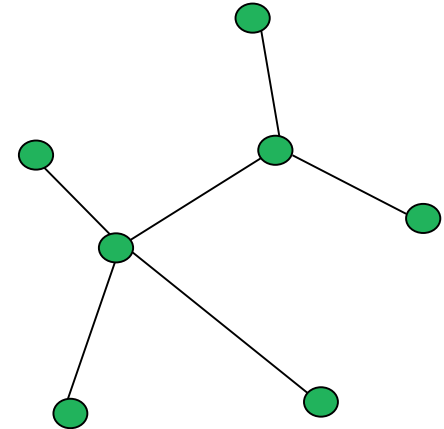
$k$ サーバ問題の競合比は $k$ 以上。

2サーバ問題の競合比は2。

限られた要求点(火事になる家)の場合、

$k$ サーバ問題の競合比は $k$ 。

- ・ライン上
- ・木上
- ・要求点が $k+1$ 個しかない場合。
- ・要求点が $k+2$ 個しかない場合。

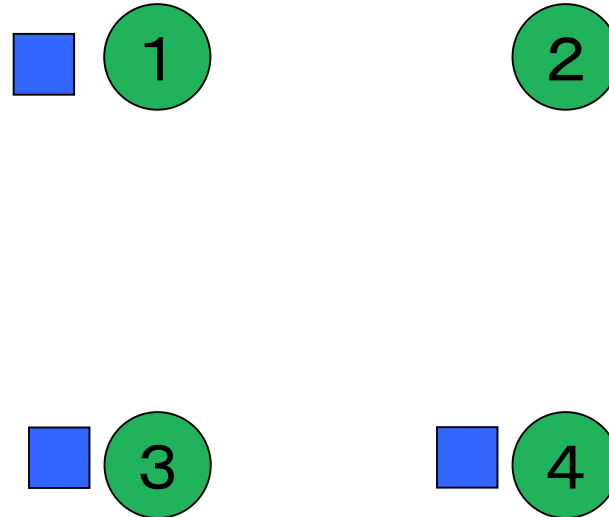


$k+1$ 個の要求点を用意

● 要求点

■ サーバ

$k=3$ の例



どんなアルゴリズムでも、比が $k$ になってしまう入力列があることを示す。(つまり、要求点が $k+1$ 個しかなくても難しい。) アルゴリズムの動きに合わせて、「意地悪い」入力列を構築していく。

● 要求点  
■ サーバ

サーバが $k$ 個で、要求点が $k+1$ 個。  
サーバがない要求点がある。  
意地悪い入力列では、常にそこを要求する。



$i$ 番目の要求の来た要求点を、 $r_i$  とする。

今の例だと、 $r_1$   $r_2$   $r_3$   $r_4$   $r_5$        $d(x, y)$  :  $x$  と  $y$  の距離

2   4   1   2   3

アルゴリズムのコスト  $\geq$

$$d(r_1, r_2) + d(r_2, r_3) + d(r_3, r_4) + \dots + d(r_{n-1}, r_n)$$

これはそんなに自明ではないので、良く考えてみよう。

$$r_2 \rightarrow r_1$$

$$r_3 \rightarrow r_2$$

$$r_4 \rightarrow r_3$$

$$r_5 \rightarrow r_4$$

と動いている。

アルゴリズムに都合の悪い入力が出来たが、それにつられて最適解も悪くなってはダメ。

最適解は、これより少なくとも $k$ 倍良いことを、これから示す。

複数( $k$ 個)のアルゴリズムを同時に走らせる。  
それらのうち最良のものは、上記の性質を満たす。

これら $k$ 個のアルゴリズムは、動作ルールは皆同じ。  
 $k$ 個のサーバの初期配置が違うだけ。

$i$ ステップ目で $r_i$  が要求されたとき、  
既にそこにサーバがあれば、何もしない。  
なければ、 $r_{i-1}$  から動かす。

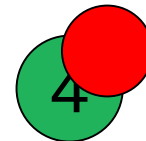
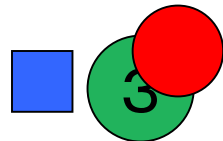
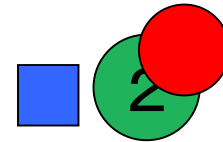
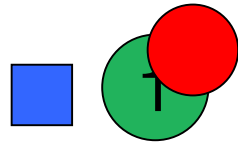
※ただし、初期配置では、 $r_1$  にはサーバがあるものとする。

例

● 要求点

■ サーバ

要求: 2 4 1 2 3





サーバが $k$ 個で、要求点が $k+1$ 個。  
「状態」は、どこが空いているかで決まる。

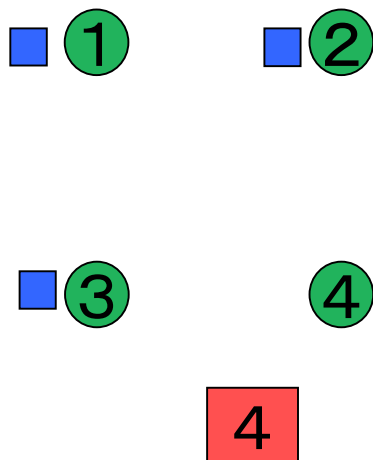


初期配置は $k+1$ 通りあり得る。  
しかし、 $r_1$ には必ずサーバがあることにしているので、  
初期配置は $k$ 通り。

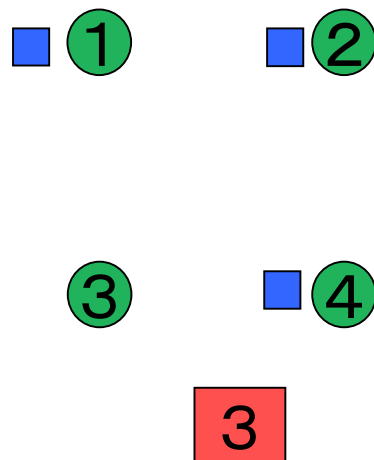
■ は状態  
(サーバのない場所)

要求: 2 4 1 2 3

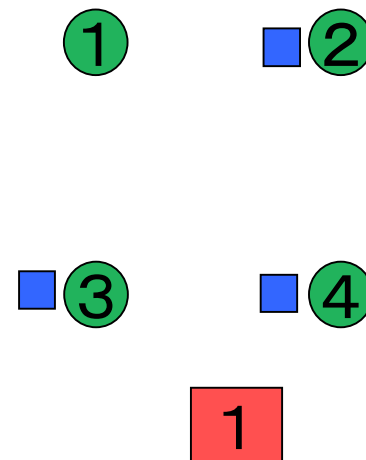
アルゴリズム1



アルゴリズム2

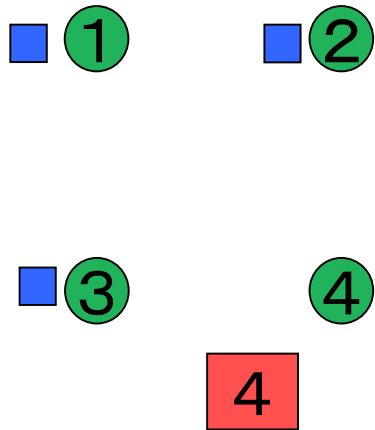


アルゴリズム3

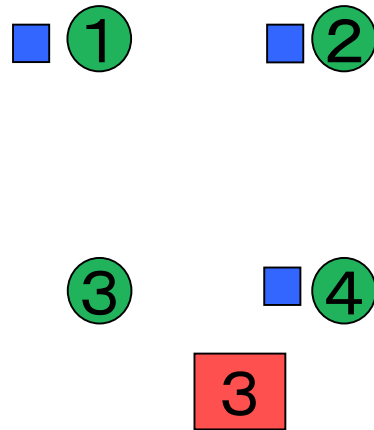


要求: 2 4 1 2 3

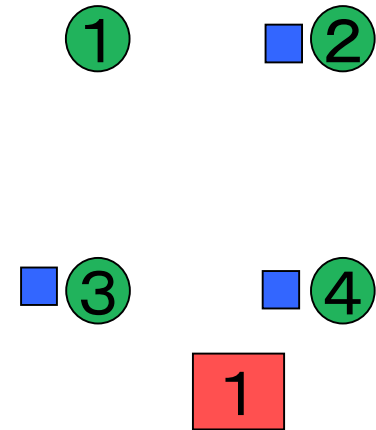
アルゴリズム1



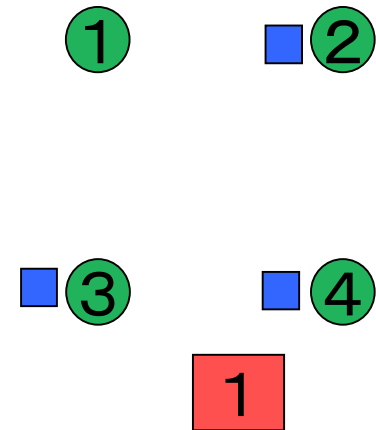
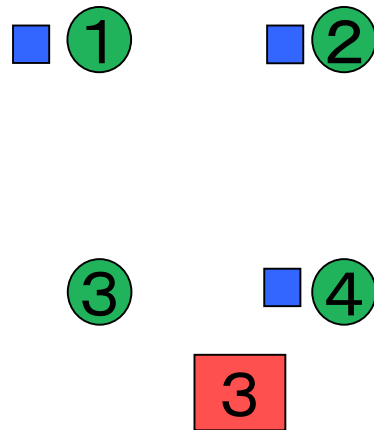
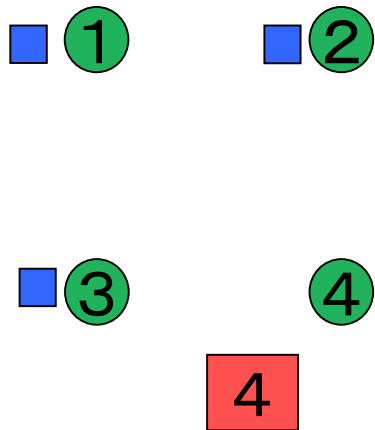
アルゴリズム2



アルゴリズム3

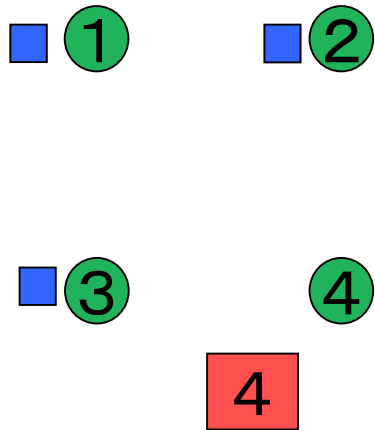


↓ 要求2

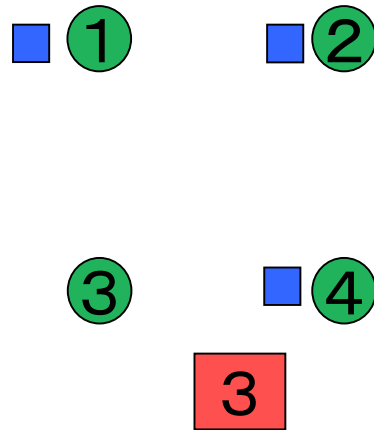


要求: 2 (4) 1 2 3

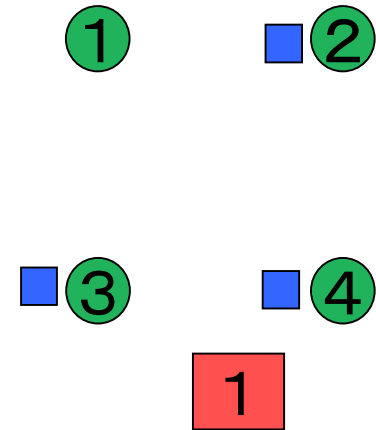
アルゴリズム1



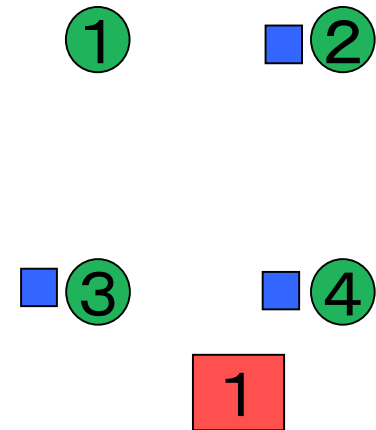
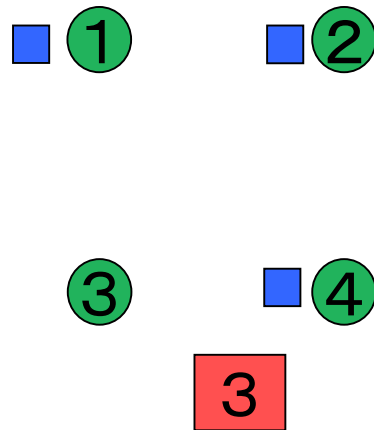
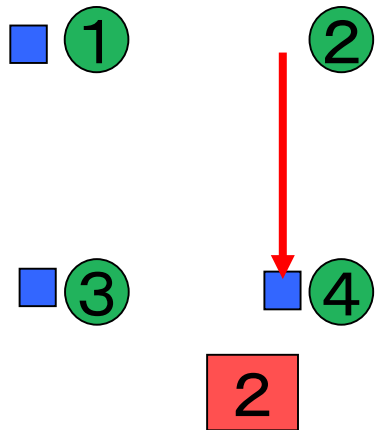
アルゴリズム2



アルゴリズム3

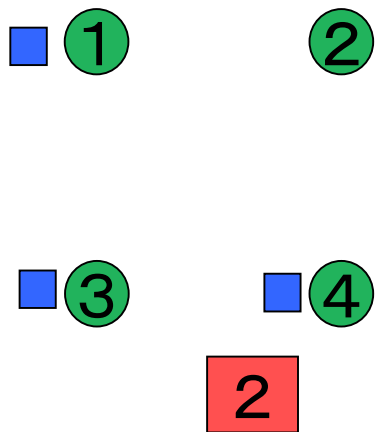


要求4

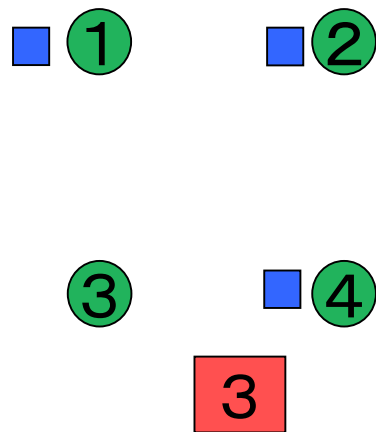


要求: 2 4 (1) 2 3

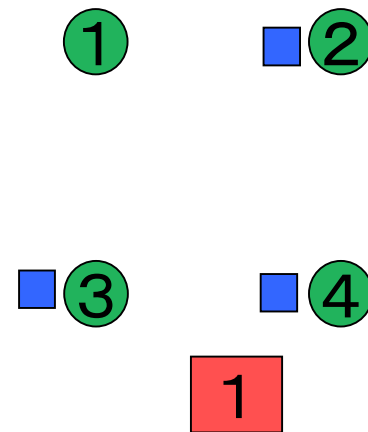
アルゴリズム1



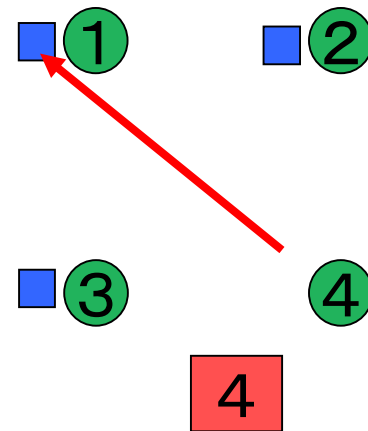
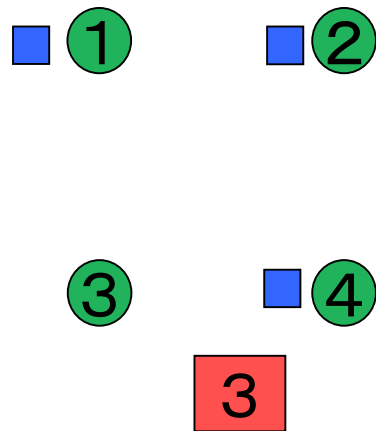
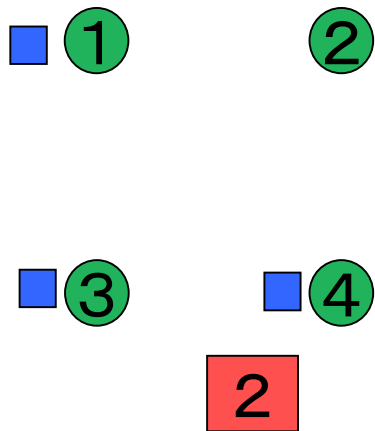
アルゴリズム2



アルゴリズム3

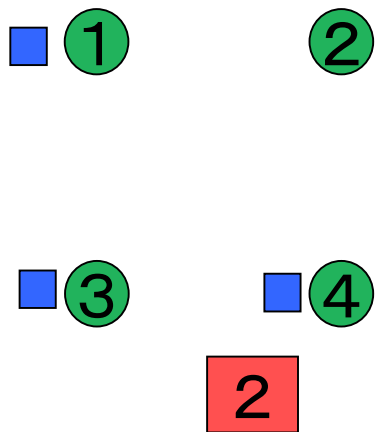


↓ 要求1

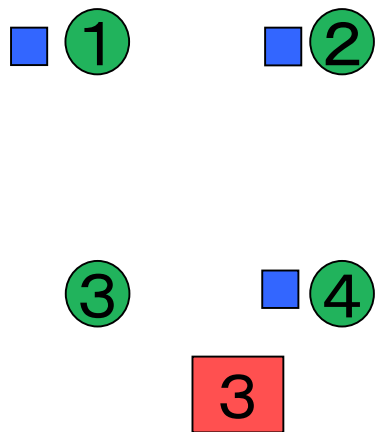


要求: 2 4 1 (2) 3

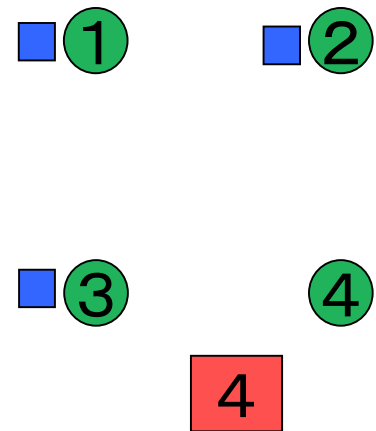
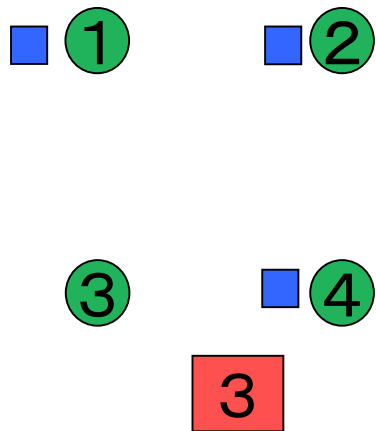
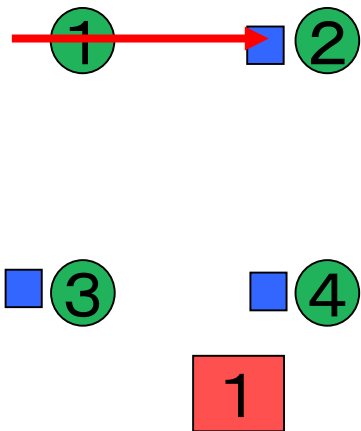
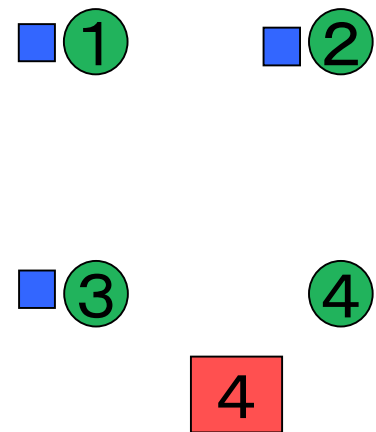
アルゴリズム1



アルゴリズム2

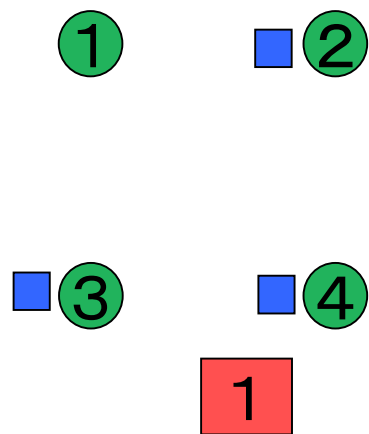


アルゴリズム3

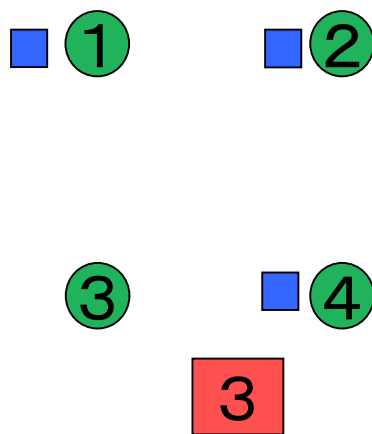


要求: 2 4 1 2 (3)

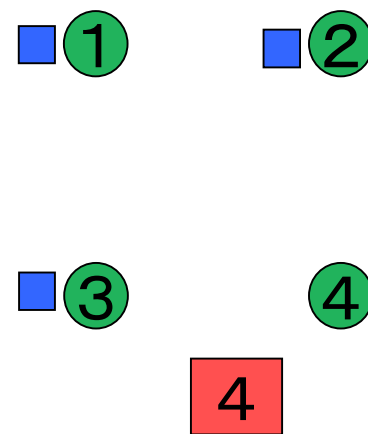
アルゴリズム1



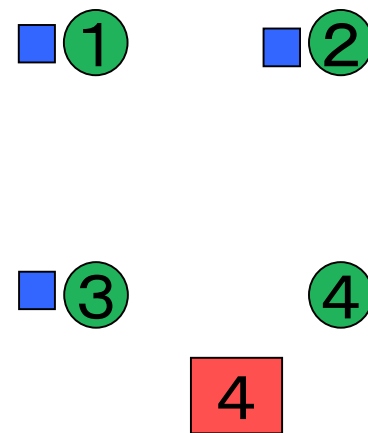
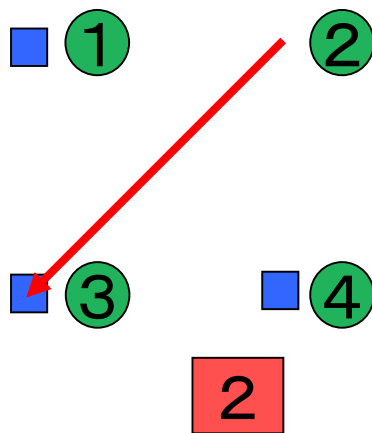
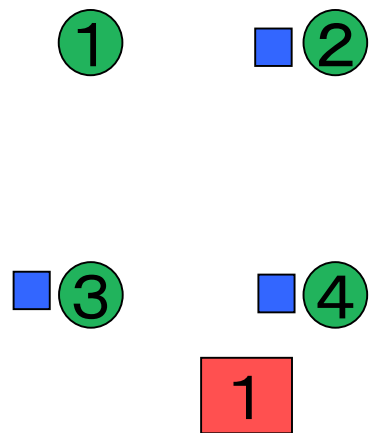
アルゴリズム2



アルゴリズム3



↓ 要求3



## 補題

どの時点においても、アルゴリズムの状態は全て異なる。

### (証明)

全てのアルゴリズムは、初期状態が違っている。



「初期状態が違う2つのアルゴリズムは、任意の時点において状態が違う」を示せば、補題の証明は完成する。

### (その証明)

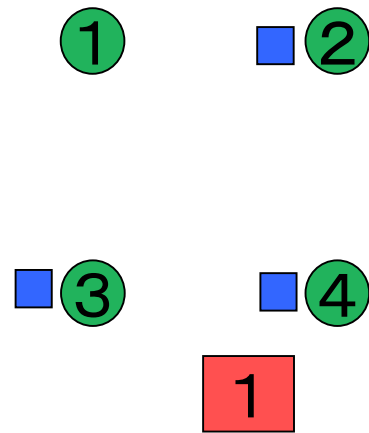
初期状態において違うので、第1リクエスト後も違う。

(初期状態において、第1リクエストの来るところには、必ずサーバがあるから。第1リクエスト後は初期状態と同じ。)

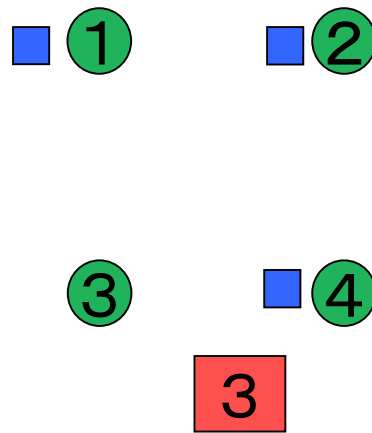
第  $i$  リクエスト後に状態が違おうとしよう。  
第  $i+1$  リクエスト後にも違おうということを、これから示す。

例えば、第  $i$  リクエスト後が以下のようになっていたとしよう。  
次のリクエストがどこに来るかで場合分けする。

アルゴリズムA



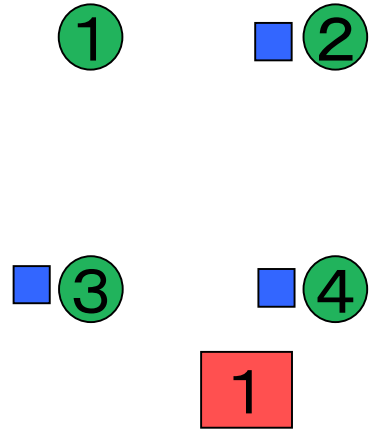
アルゴリズムB



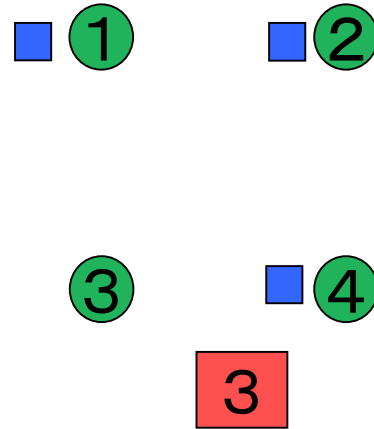
- (1) AもBもサーバのある要求点(例えば上記の2)に来る。  
AもBも動かさない。  
→  $i+1$  回目のリクエスト後も、状態は異なる。



## アルゴリズムA

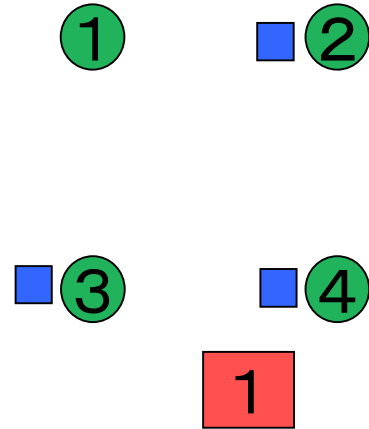


## アルゴリズムB

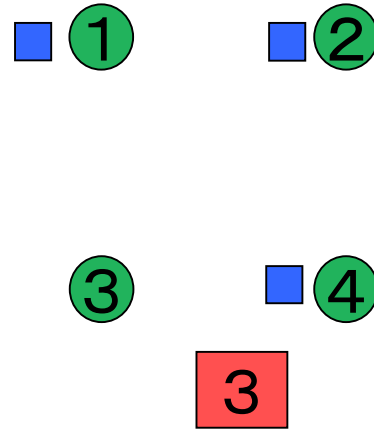


(2) AもBもサーバのない要求点に来る。  
そんな要求点はない。

## アルゴリズムA



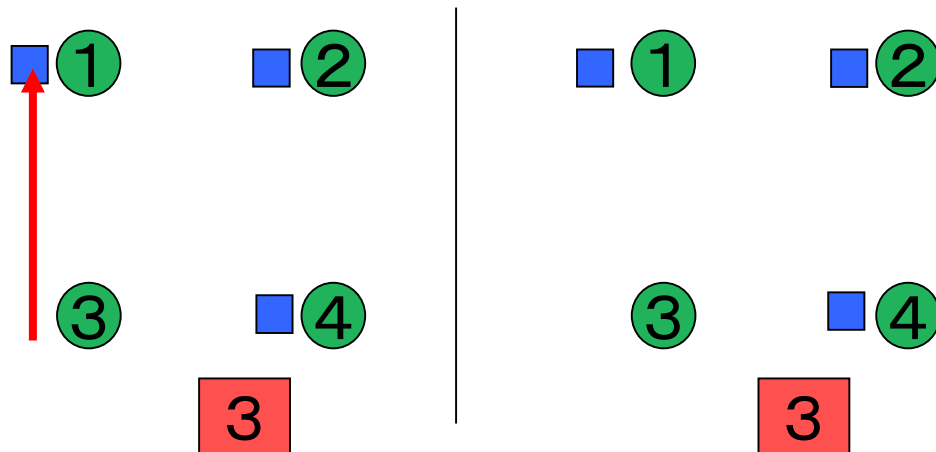
## アルゴリズムB



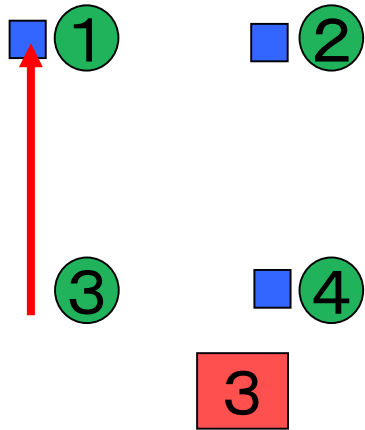
(3) Aではサーバがないが、Bではサーバのある要求点  
(上記の例では1)に来る。

→ Aは動かして、Bは動かさない。

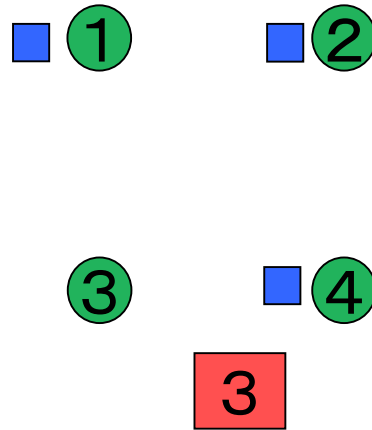
困るのは、上記の例で、Aが3から1に動かす。



## アルゴリズムA



## アルゴリズムB



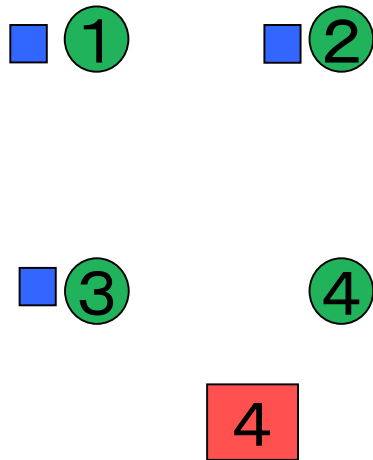
しかし、それなら、ルールより、第  $i$  リクエストの要求は3だった。  
しかし、Bは第  $i$  リクエスト後には要求点3にサーバを置いていない。  
→ 矛盾

**(証明終わり)**

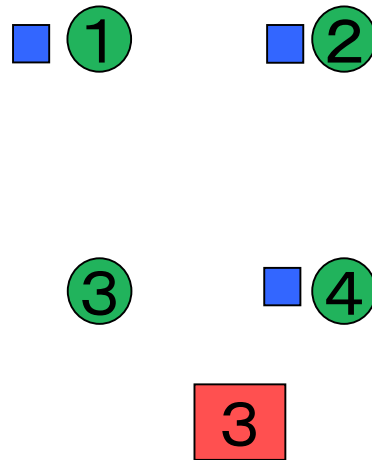
これら $k$ 個のアルゴリズムを同時に動かす。  
そして、総移動距離を考える。

要求: (2) (4) (1) (2) (3)

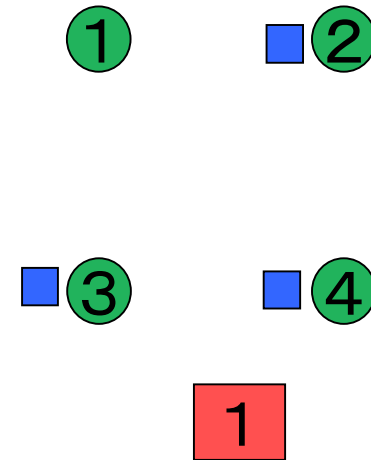
アルゴリズム1



アルゴリズム2



アルゴリズム3



各ステップにおいて、全てのアルゴリズムの状態が異なるから、  
サーバを動かすアルゴリズムは1つだけ。  
しかもその距離は、ルールより、 $d(r_{i-1}, r_i)$

$k$ 個のアルゴリズムのコストの総和:

$$d(r_1, r_2) + d(r_2, r_3) + d(r_3, r_4) + \dots + d(r_{n-1}, r_n)$$



少なくとも1つは、コストの総和が

$$\frac{1}{k} (d(r_1, r_2) + d(r_2, r_3) + d(r_3, r_4) + \dots + d(r_{n-1}, r_n))$$

以下。OPTは当然それ以下。

アルゴリズムのコスト  $\geq$

$$d(r_1, r_2) + d(r_2, r_3) + d(r_3, r_4) + \dots + d(r_{n-1}, r_n)$$

したがって、最適解は今考えているアルゴリズムより、少なくとも $k$ 倍は良い。

これで、任意のアルゴリズムの競合比が $k$ 以上であることが示せた。

## 有名な予想 ( $k$ サーバ予想)

「 $k$ サーバ問題の競合比は $k$ 。」

現段階で知られていること。

$k$ サーバ問題の競合比は $2k-1$ 以下。

$k$ サーバ問題の競合比は $k$ 以上。

2サーバ問題の競合比は2。

限られた要求点(火事になる家)の場合、

$k$ サーバ問題の競合比は $k$ 。

・ライン上

・木上

・要求点が $k+1$ 個しかない場合。

・要求点が $k+2$ 個しかない場合。

## アルゴリズム Double Coverage (DC)

ルール: 左端のサーバより左に要求が来たら

→ 左端のサーバを動かす

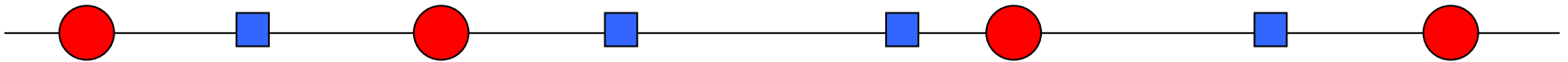
右端のサーバより右に要求が来たら

→ 右端のサーバを動かす

2つのサーバの間に要求が来たら

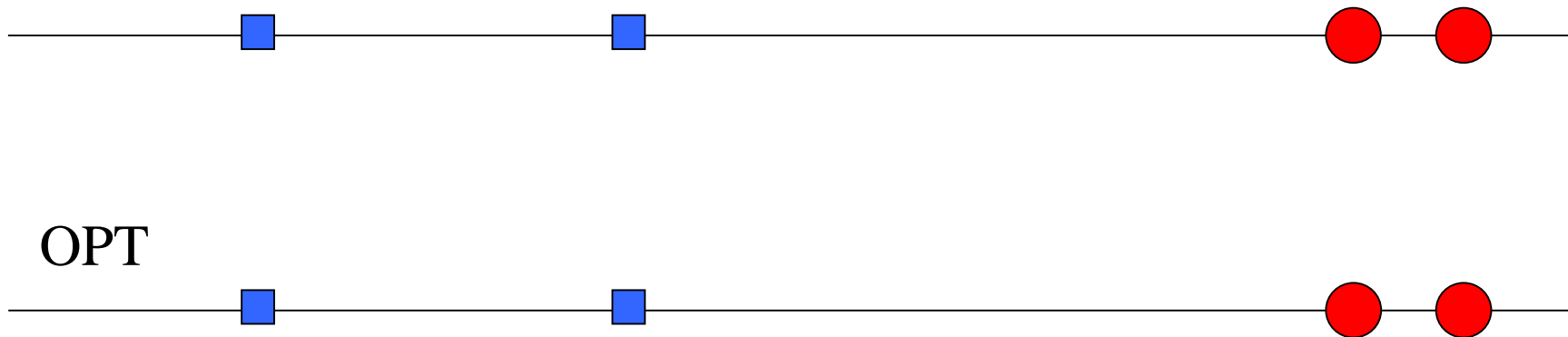
→ それら2つのサーバを同じ距離だけ動かす

$k=4$ の例



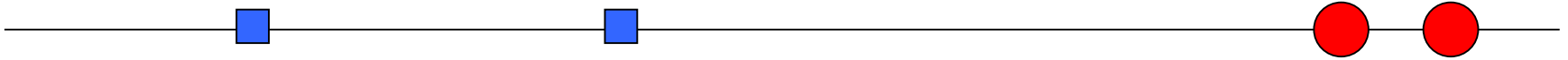
定理: DCの競合比は  $k$  以下である。

貪欲アルゴリズムの競合比は、ライン上でも無限大





## 同じ入力にDouble Coverageだと



左のサーバがちょっとずつ近づいてくるので、  
右のサーバが「無限に振らされる」ということにはならない。

定理: DCの競合比は  $k$  以下である。

証明に入る前に

解析の基本的な考え方 ... **ならし解析 (Amortized Analysis)**

理想的な状況

1ステップで、

OPTのコストが  $c$  → アルゴリズムのコストが  $kc$  以下

これが言えれば、競合比が  $k$  以下であることは言える。

しかし、この条件は強すぎる。

例えば

	OPT	アルゴリズム
ステップ1	5	8
ステップ2	2	12
・	11	6
・	9	10
・	2	2
ステップ6	7	10
	36	48

実際の比は $4/3$ なのに、解析では6になってしまう。  
(つまり、解析で損をしている)

# ならし解析

ならしコスト = アルゴリズムのコスト + ならし分

	OPT	アルゴリズム	ならし分	ならしコスト
ステップ1	5	8	-1	7
ステップ2	2	12	-10	2
・	11	6	10	16
・	9	10	2	12
・	2	2	0	2
ステップ6	7	10	0	10

もちろん、ならしコストをどう定義するかが難しい！

## 重要なポイント:

- ・ 毎ステップ  $\text{ならしコスト} \leq 1.5 \times \text{OPTのコスト}$
- ・ ならし分の合計  $\geq 0$

→ アルゴリズムのコストの合計  $\leq$  ならしコストの合計  $\leq 1.5 \times \text{OPTのコストの合計}$

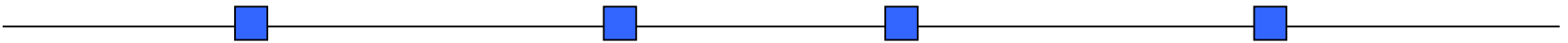
これから、「アルゴリズムの競合比  $\leq 1.5$ 」が言える。

定理: DCの競合比は  $k$  以下である。

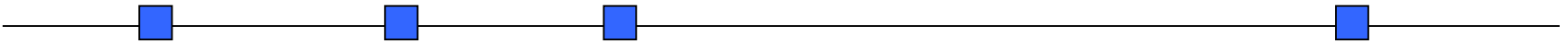
(証明)

ある時点において、

DCの配置



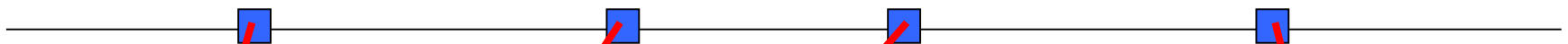
OPTの配置



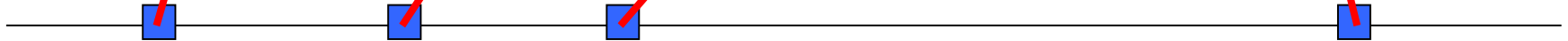
このとき、DCとOPT間のサーバ同士の、最小マッチングの重みを  $M$  とする。

最小重みマッチングは、左から順に対応させることにより、与えられる。  
→ 線は交差しない。

DCの配置



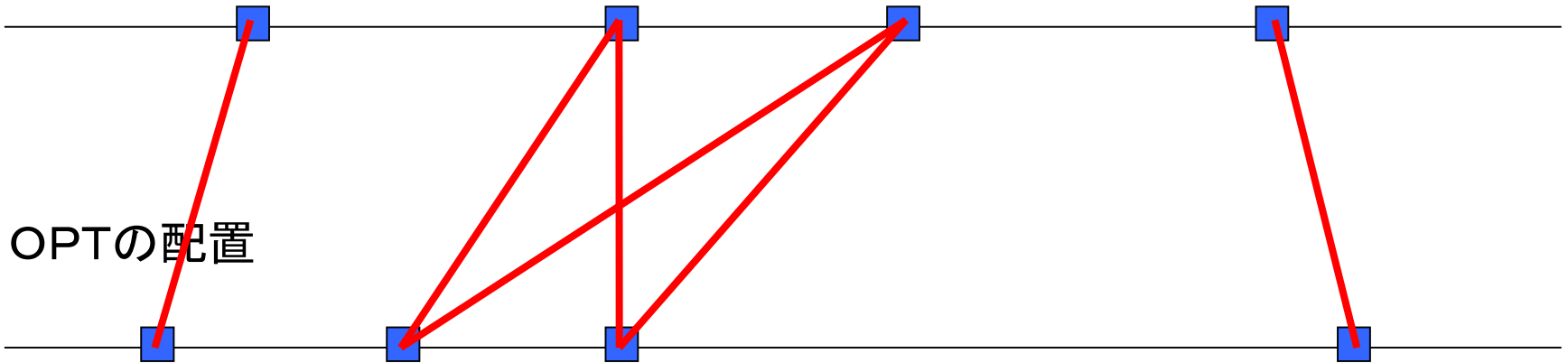
OPTの配置



(見やすいように、上下で対応させているが、実際の「距離」は直線上で測る)

なぜなら、もし交差していたら、交差を解消することで、マッチングの重みは必ず減るから。

DCの配置



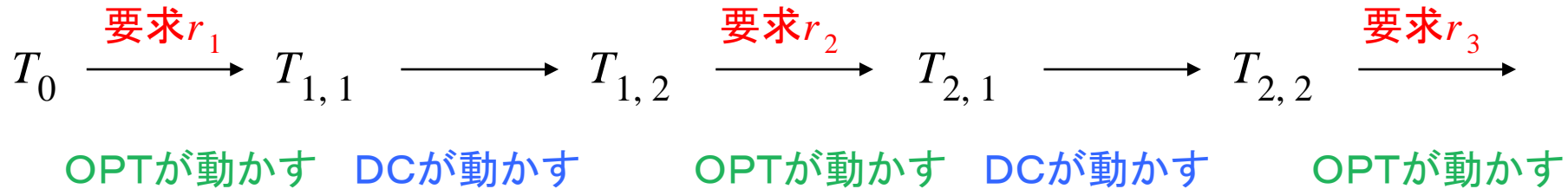
OPTの配置

ある時点でのポテンシャル関数 (OPTとDCの配置のみから決まる値) を  $T = kM + \Sigma$  とする。

$\Sigma$  は、DCにおける、2つのサーバ間の距離の総和。  $\rightarrow \binom{k}{2}$  個の和

ポテンシャル関数の変化を解析する。

$$T = kM + \Sigma$$



(i)  $T_{i,1} - T_{(i-1),2} \leq kd_i$        $d_i$ はこの要求に対するOPTのコスト

OPTが動かす番なので、 $\Sigma$ は変化しない。

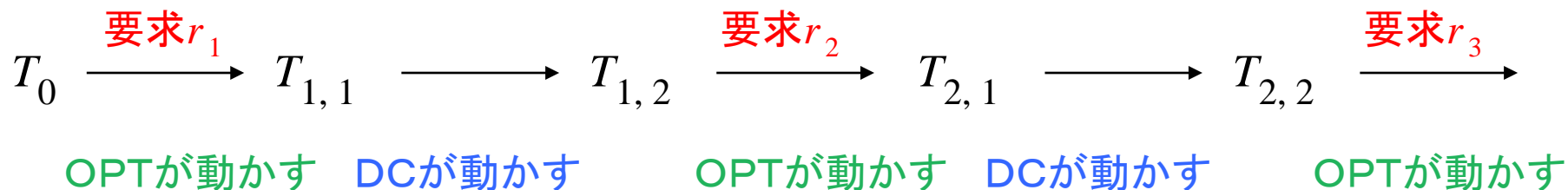
$M$ は、マッチングは元のままだとしても、OPTが $d_i$ しか動かないので、 $d_i$ しか増えない。よって全体で $kd_i$ しか増えない。

(ポテンシャル関数は、 $M$ に $k$ が掛かっていることに注意！)



ポテンシャル関数の変化を解析する。

$$T = kM + \Sigma$$



$$(ii) T_{i,2} - T_{i,1} \leq -s_i$$

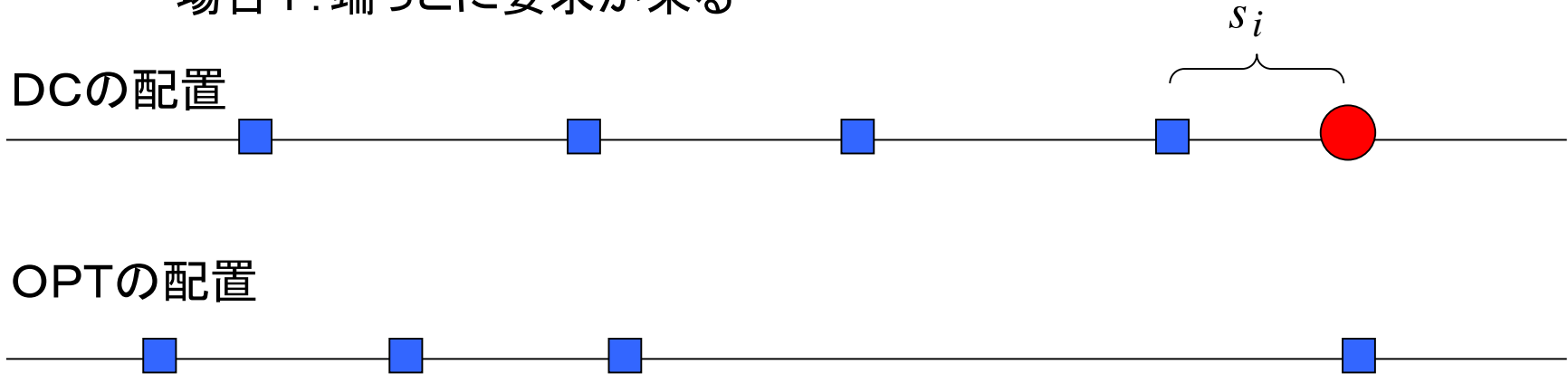
$s_i$  はこの要求に対するDCのコスト

つまり、少なくとも  $s_i$  は減る

$$(ii) T_{i,2} - T_{i,1} \leq -s_i$$

$s_i$  はこの要求に対するDCのコスト

場合1: 端っこに要求が来る



OPTはそこに既にサーバがある(OPTが動作した後の状態だから)

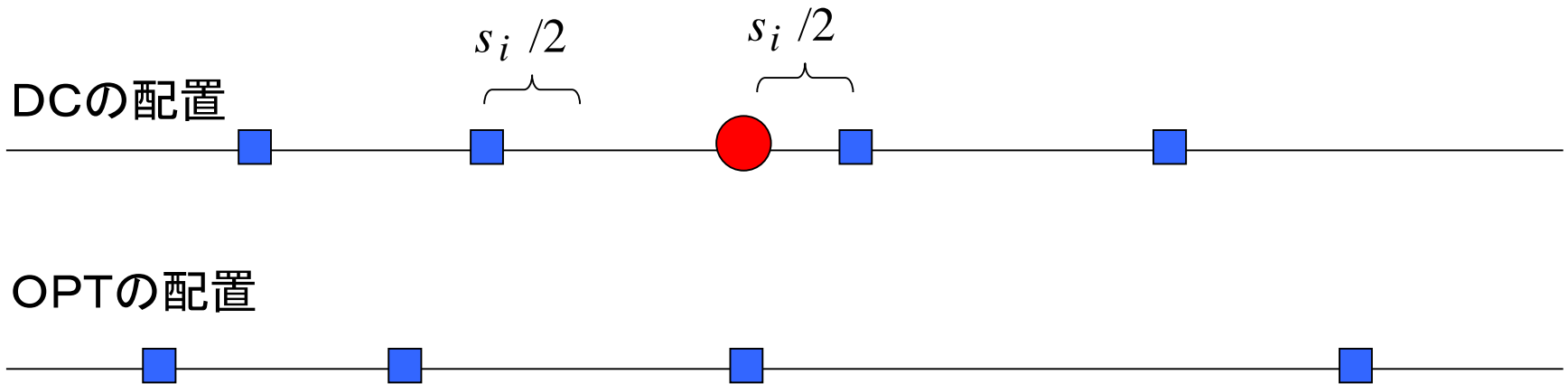
- ・ $M$  は  $s_i$  減る (DCの動かしたサーバは、一番右だった。OPTは、少なくとも要求点にはサーバがいる。もしくは、それより右にもいるかもしれない。今動かしたDCのサーバは、とにかく要求点より右の(OPTの)サーバとマッチしているはず。)
- ・ $\Sigma$  は  $(k-1)s_i$  増加する。(自分以外の  $k-1$  個のサーバとの距離が、それぞれ  $s_i$  ずつ増えるから。)

↓  
全体として  $s_i$  減る。

$$(ii) T_{i,2} - T_{i,1} \leq -s_i$$

$s_i$  はこの要求に対するDCのコスト

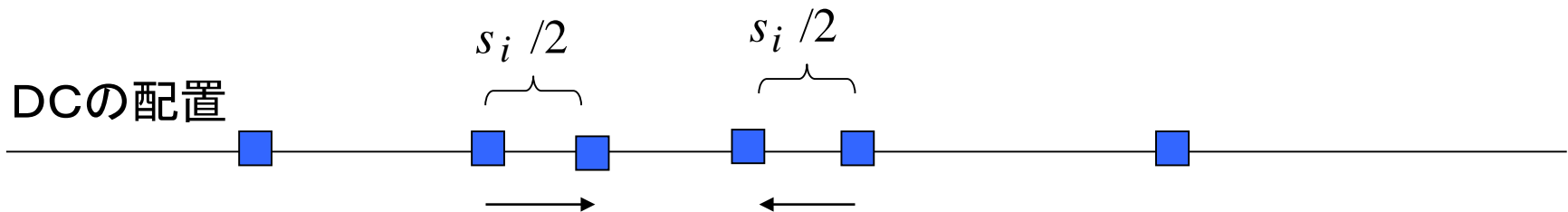
場合2:間に要求が来る



$$(ii) T_{i,2} - T_{i,1} \leq -s_i$$

$s_i$  はこの要求に対するDCのコスト

場合2:間に要求が来る

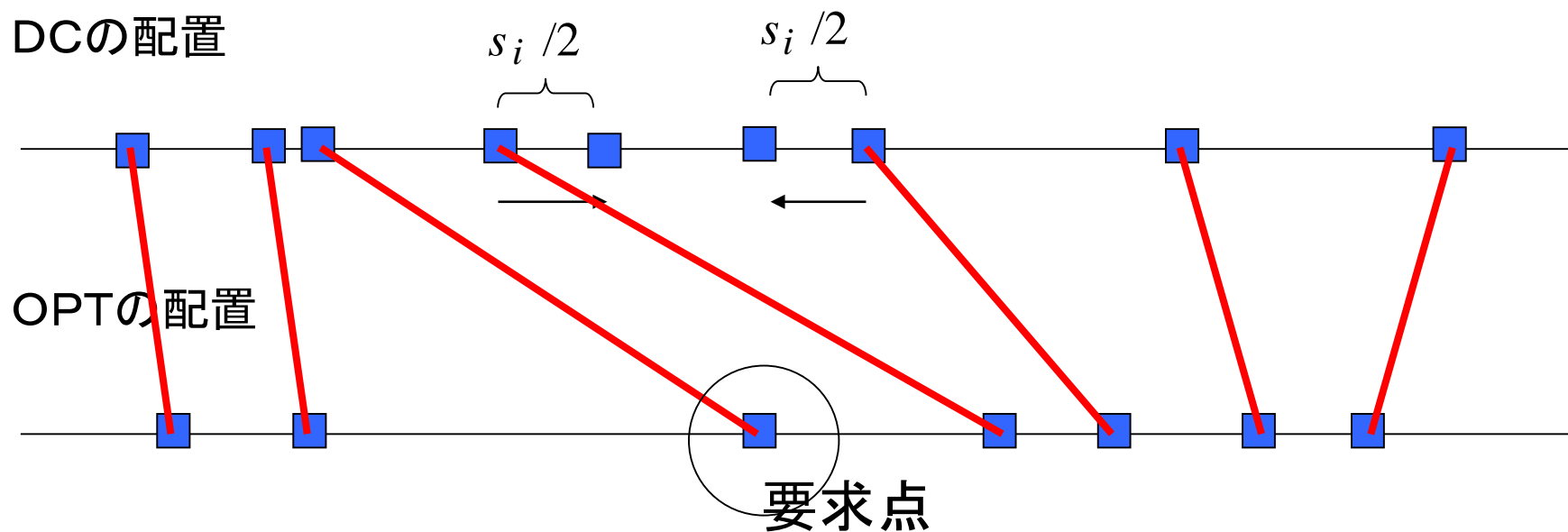


- $\Sigma$  は、動いた2つのサーバ間は  $s_i$  減る。  
それ以外は増えも減りもしない。  
(例えば、左にあるサーバだと、1個が  $s_i/2$  遠のいて、もう一個が  $s_i/2$  近づく。つまり、それ対他のサーバの距離の総和は変わらない。)

- $M$  は、動かした2個のうち、1個に関連して必ず  $s_i/2$  は減る。  
もう1個に関連する分は、増えたとしても高々  $s_i/2$ 。

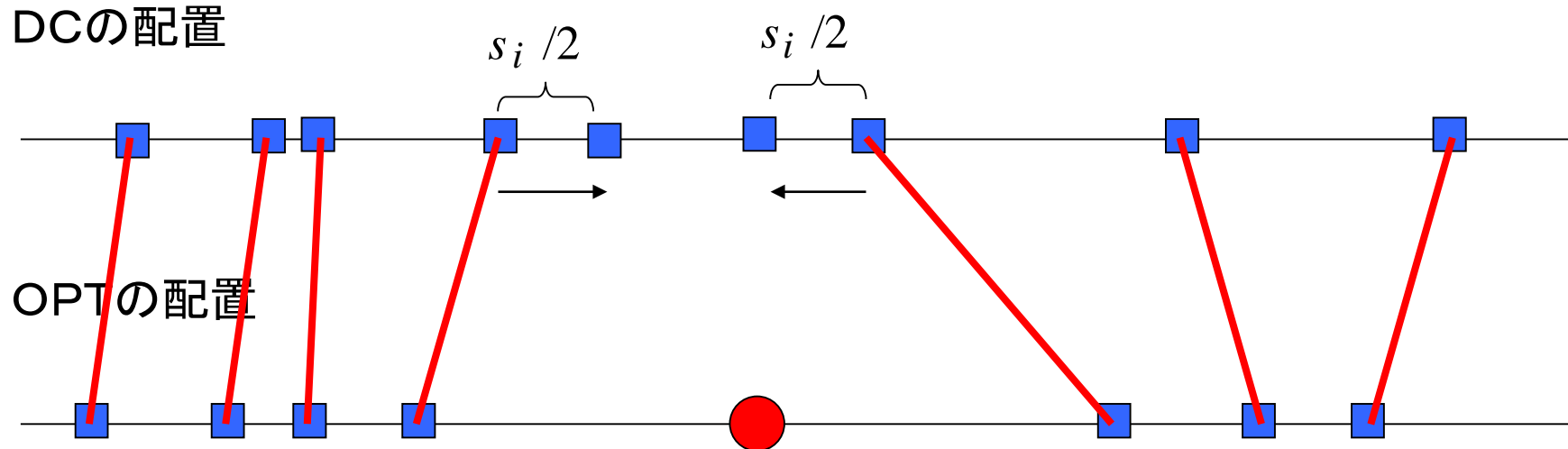
次ページで、もう少し詳しく見る。

DCの配置



- ・ $M$  は、動かした2個のうち、1個に関連して必ず  $s_i/2$  は減る。  
もう1個に関連する分は、増えたとしても高々  $s_i/2$ 。

DCの配置

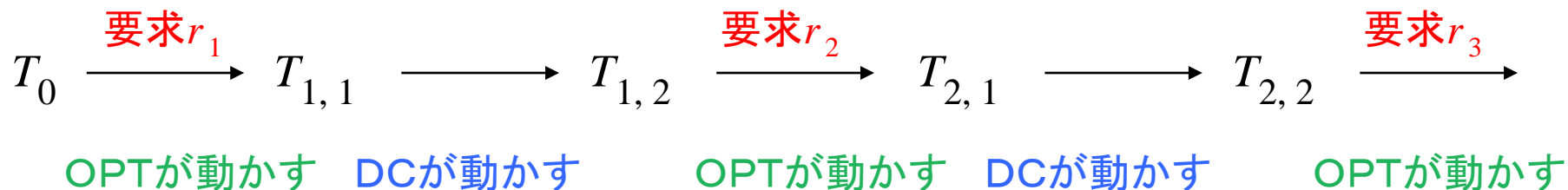


OPTの配置

逆に、両方とも増えらしたら、こんな感じ。  
でも、OPTの方が要求点にサーバがないことになるので矛盾。

## ここまで証明したことのまとめ

ポテンシャル:  $T = kM + \Sigma$



(i)  $T_{i,1} - T_{(i-1),2} \leq kd_i$

$d_i$  はこの要求に対するOPTのコスト

(ii)  $T_{i,2} - T_{i,1} \leq -s_i$

$s_i$  はこの要求に対するDCのコスト

(i)と(ii)を足すと

$$T_{i,2} - T_{(i-1),2} \leq kd_i - s_i$$

$$s_i + \boxed{T_{i,2} - T_{(i-1),2}} \leq kd_i$$

↑  
ならし分

$$s_i + T_{i,2} - T_{(i-1),2} \leq kd_i$$



$i$ ステップ目でのならしコスト

$i$ ステップ目でのOPTのコストの $k$ 倍

ならしコストの総和

$$\begin{array}{r}
 \cancel{T_{n,2}} - \cancel{T_{(n-1),2}} \\
 \cancel{T_{(n-1),2}} - \cancel{T_{(n-2),2}} \\
 \cancel{T_{(n-2),2}} - \cancel{T_{(n-3),2}} \\
 \vdots \\
 \cancel{T_{2,2}} - \cancel{T_{1,2}} \\
 \cancel{T_{1,2}} - T_0
 \end{array}$$

---


$$T_{n,2} - T_0$$

$T_{n,2} - T_0$  は $n$ によらない定数。

本当は「ならし分の合計 $\geq 0$ 」  
を言わなければならないのだけど、  
競合比の定義ではこれでOK. (省略)

よって、競合比 $k$ が言えた。

(証明終わり)



直接的にやってみると、

$s_i + T_{i,2} - T_{(i-1),2} \leq kd_i$  を全ての*i*について足し合わせると



$$s_1 + s_2 + \cdots + s_n \leq k(d_1 + d_2 + \cdots + d_n) + T_0 - T_{n,2}$$

アルゴリズムの総コスト

OPTの総コスト