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情況で、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σを最後まで読んだ後なら、最適解を計算できる
$\rightarrow \mathrm{OPT}(\sigma)$

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

「Aの競合比がc」とは、全てのσに対して、

$$\frac{\mathrm{A}(\sigma)}{\mathrm{OPT}(\sigma)} \leqq c$$

（最小化問題の場合）

$$\frac{\mathrm{OPT}(\sigma)}{\mathrm{A}(\sigma)} \leqq c$$

（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：レンタルスキー問題

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出来る（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。
100万/5万＝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。
6万/2万＝3倍悪い。
（つまり競合比は3以上。）

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、最初に買って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。

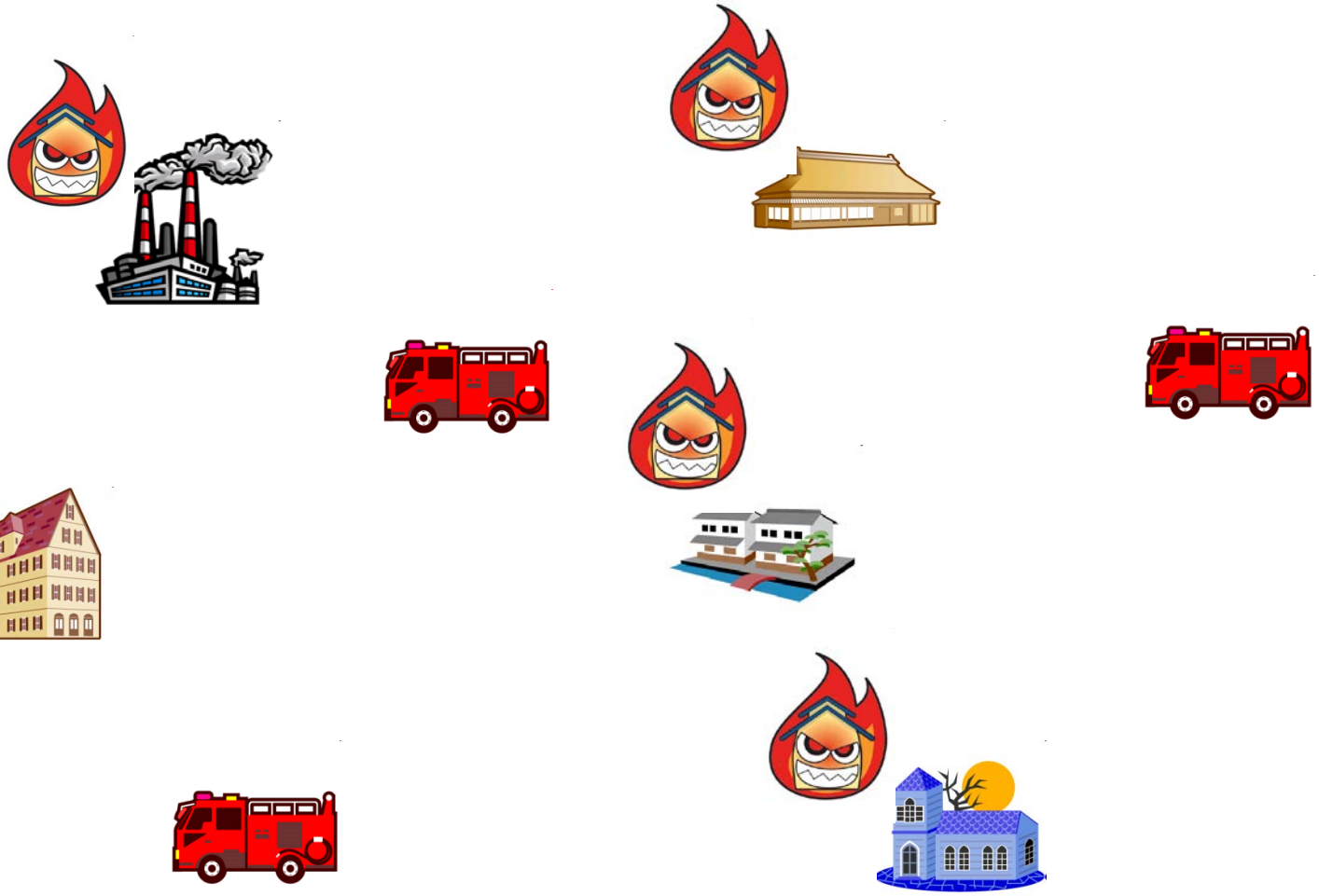
アルゴリズムとしては、一度買ったらそれ以降それを使うのがまし。
なので、「何回目に買うか」でアルゴリズムは決まる。

k-1回目までレンタルして、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が最悪。
　かかったお金は$(k-1)+5=k+4$万円
　・$1 \leqq k \leqq 4$のとき、最適解は、毎回レンタルしてk万円
　　$(k+4)/k = 1+4/k \geqq 2$
　　なので、比が1.8よりも悪くなる。
　・$k \geqq 6$のとき、最適解は、最初に買って5万円。
　　$(k+4)/5 \geqq 2$
　　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









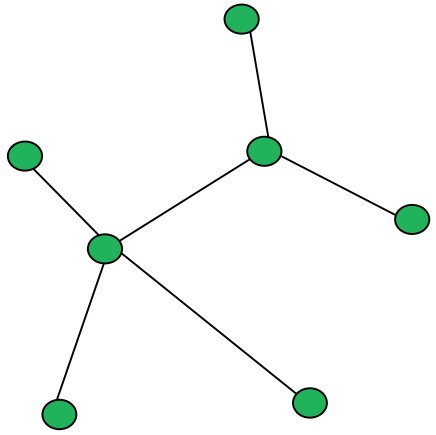


問題：貪欲アルゴリズムにとって都合の悪い入力はあるか？

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

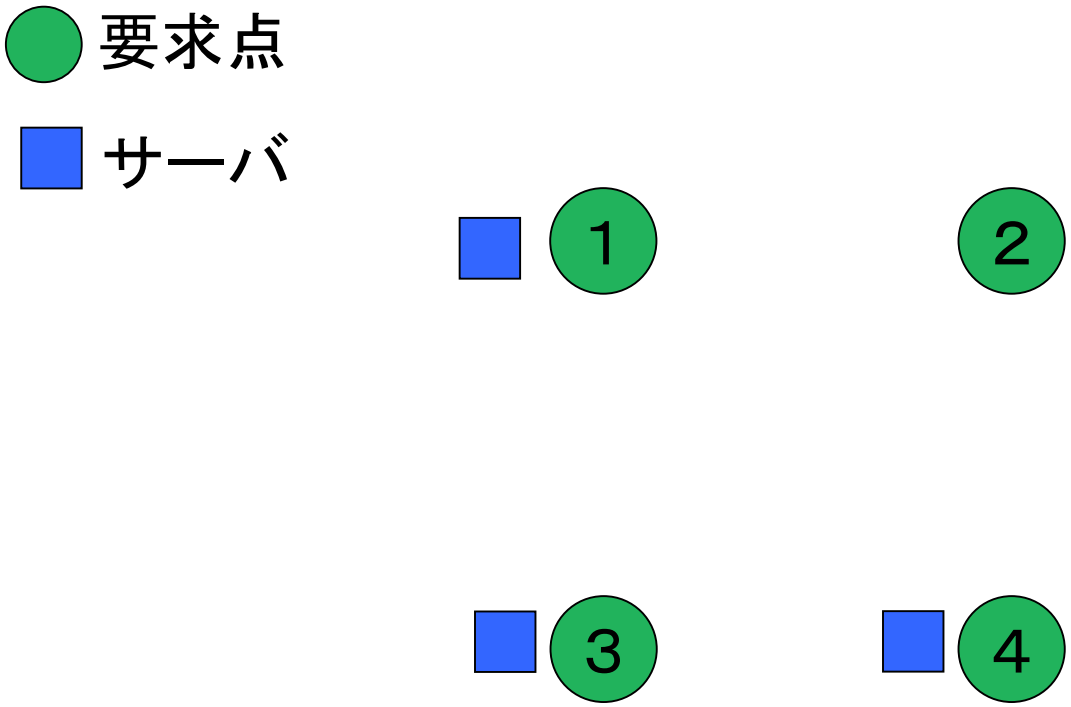
・木上

・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

k=3の例

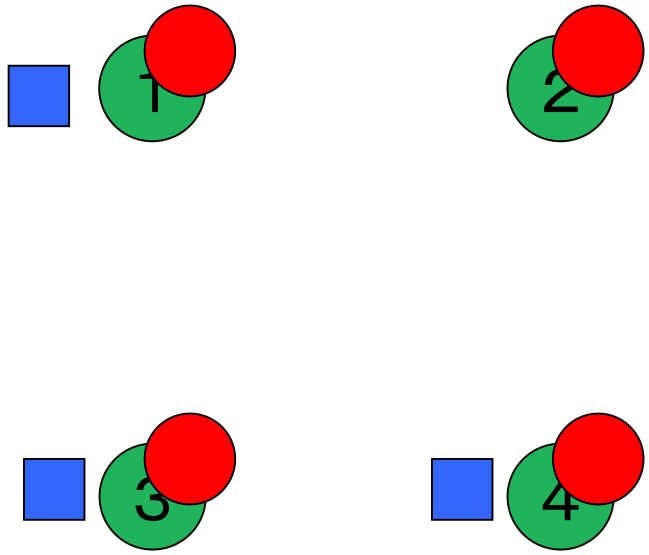


どんなアルゴリズムでも、比がk になってしまう入力列があることを示す。（つまり、要求点がk+1個しかなくても難しい。）
アルゴリズムの動きに合わせて、「意地悪い」入力列を構築していく。

要求点

サーバ

サーバがk個で、要求点が$k+1$個。
サーバがいない要求点が必ずある。
　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

２　４　１　２　３

$d(x, y)$: x と y の距離

アルゴリズムのコスト≧

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

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。

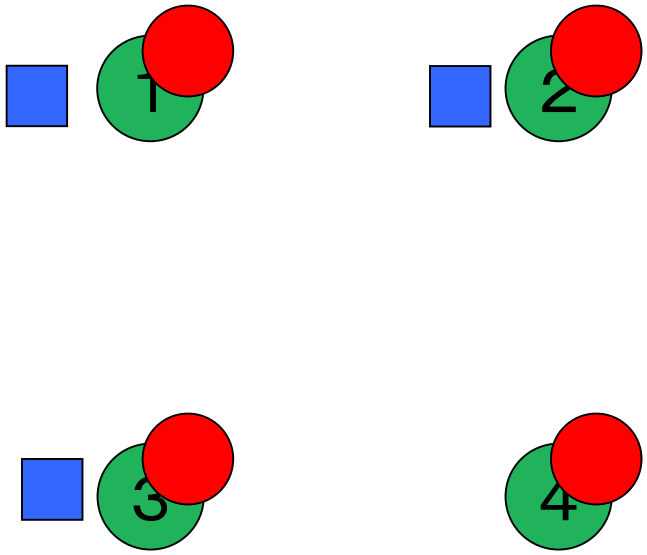
※ただし、初期配置では、r_1 にはサーバがあるものとする。

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。

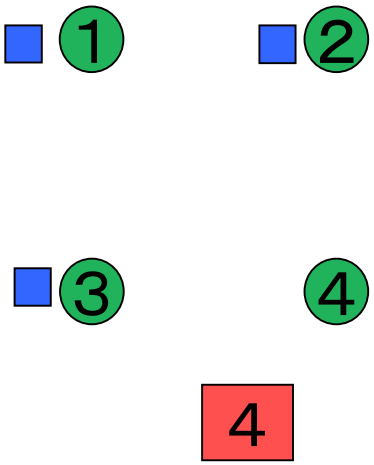
↓

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。

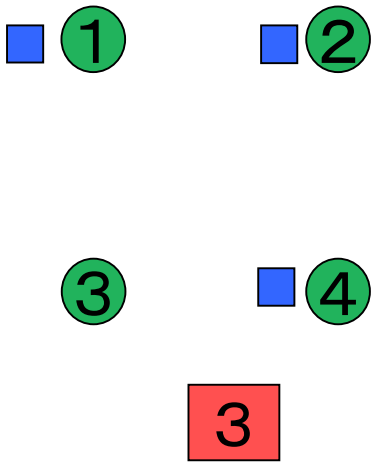
■ は状態
（サーバのない場所）

要求：　２　４　１　２　３

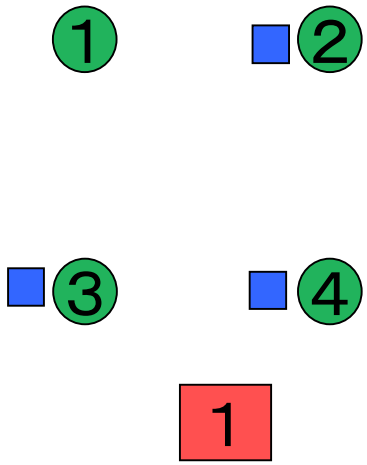
アルゴリズム１



アルゴリズム２

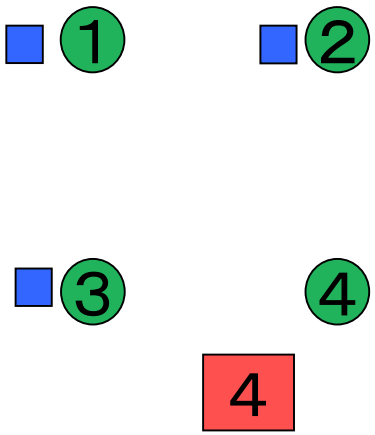


アルゴリズム３

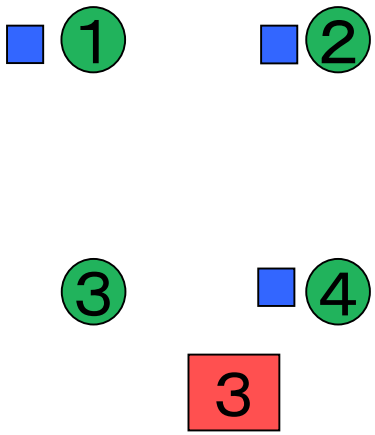


要求：2　4　1　2　3

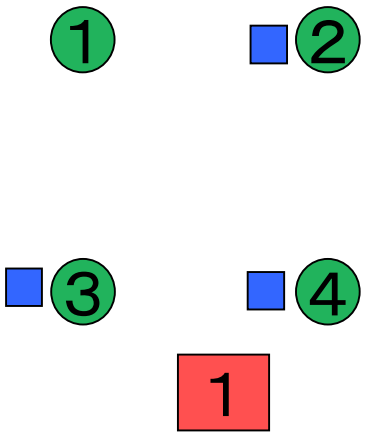
アルゴリズム1



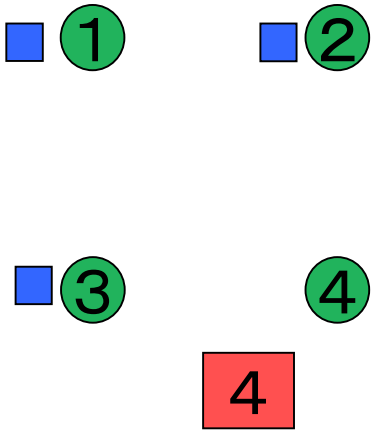
アルゴリズム2

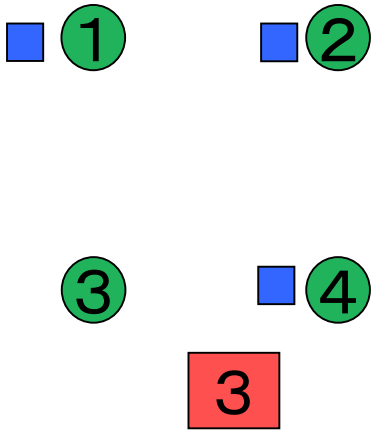


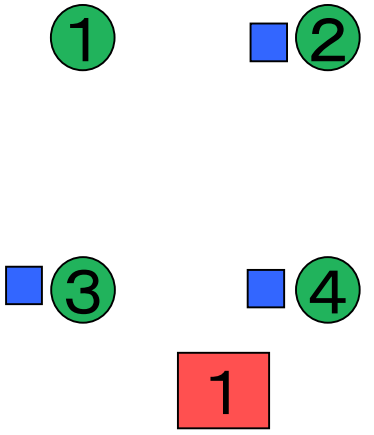
アルゴリズム3



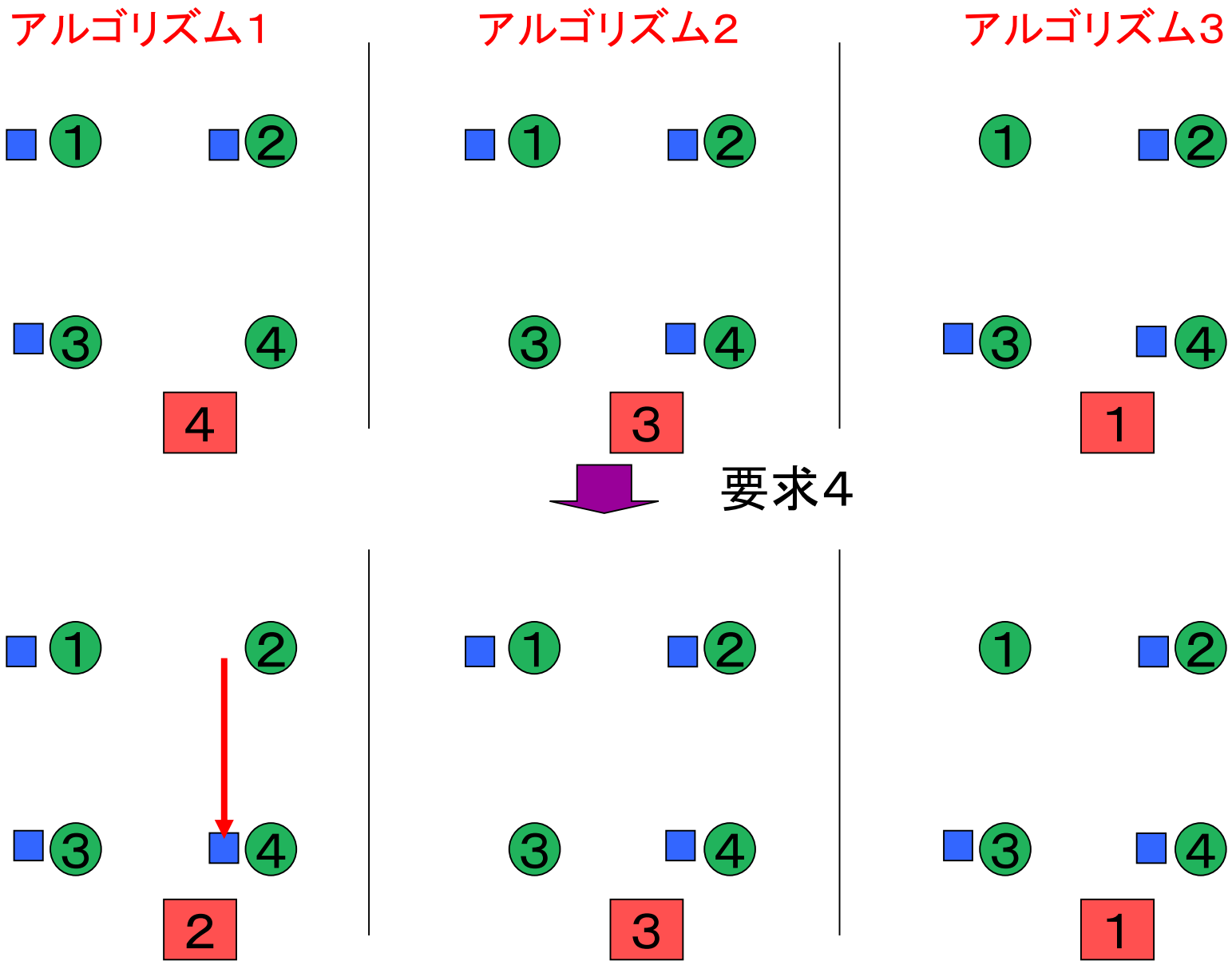
要求2



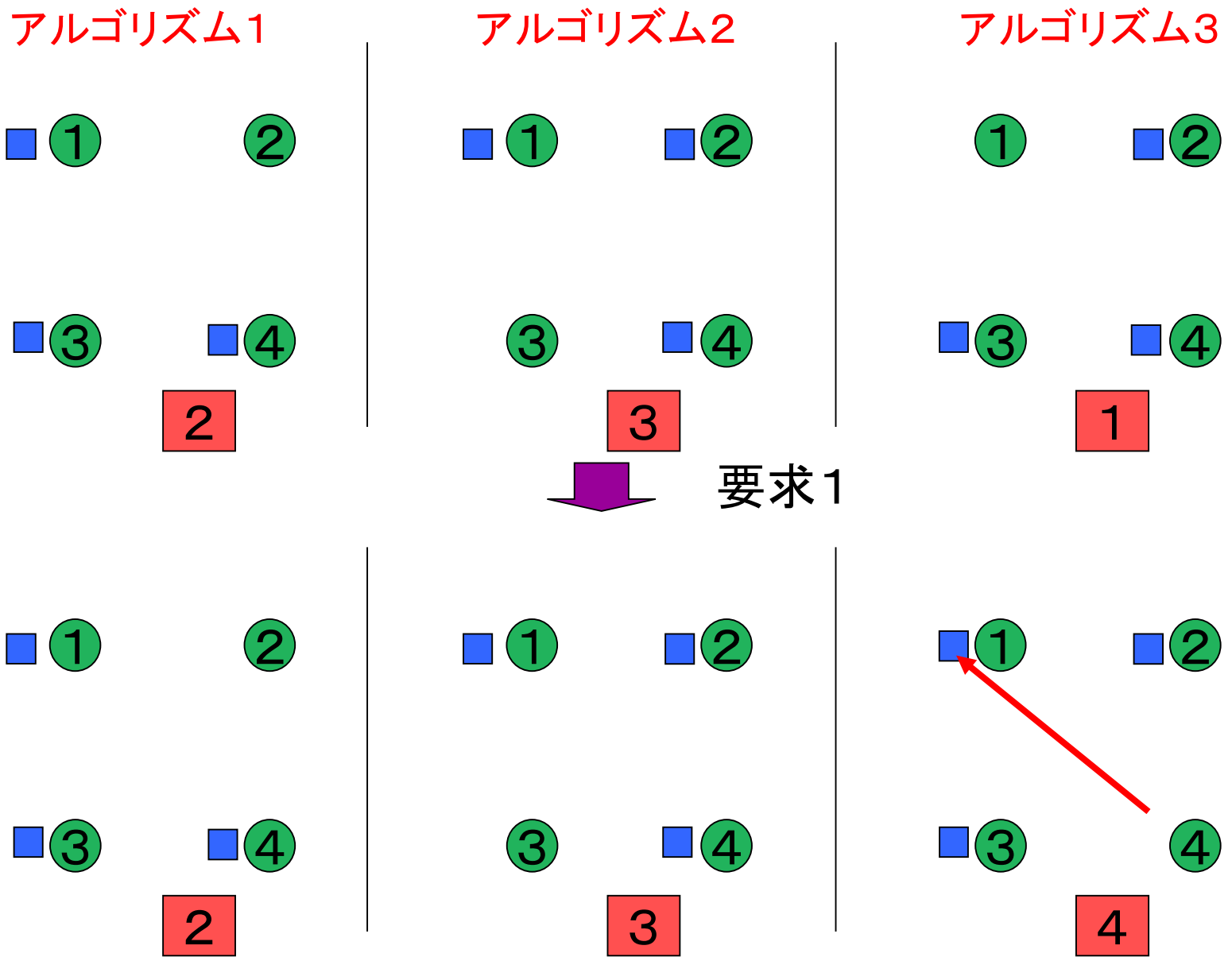




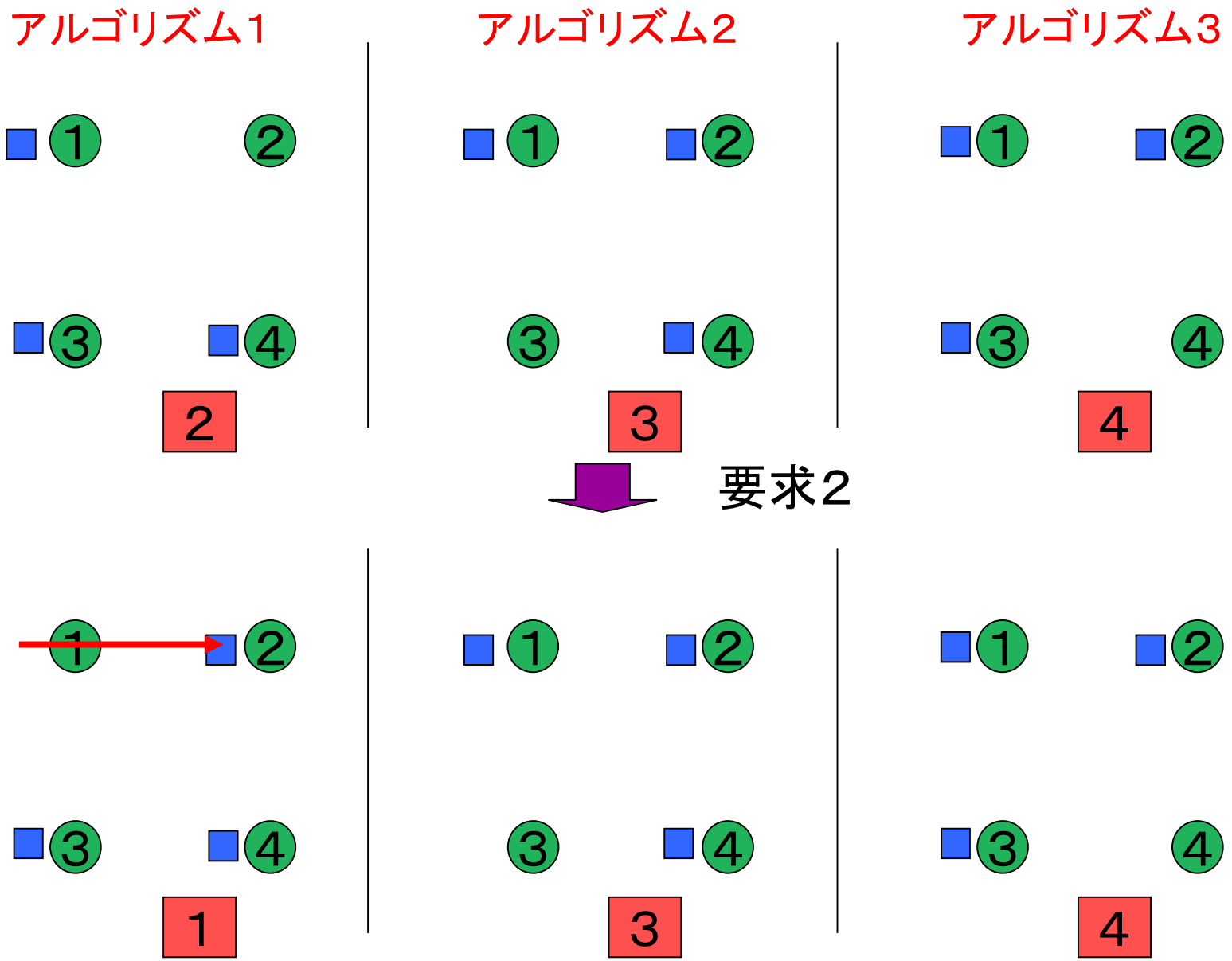
要求：　2　4　1　2　3



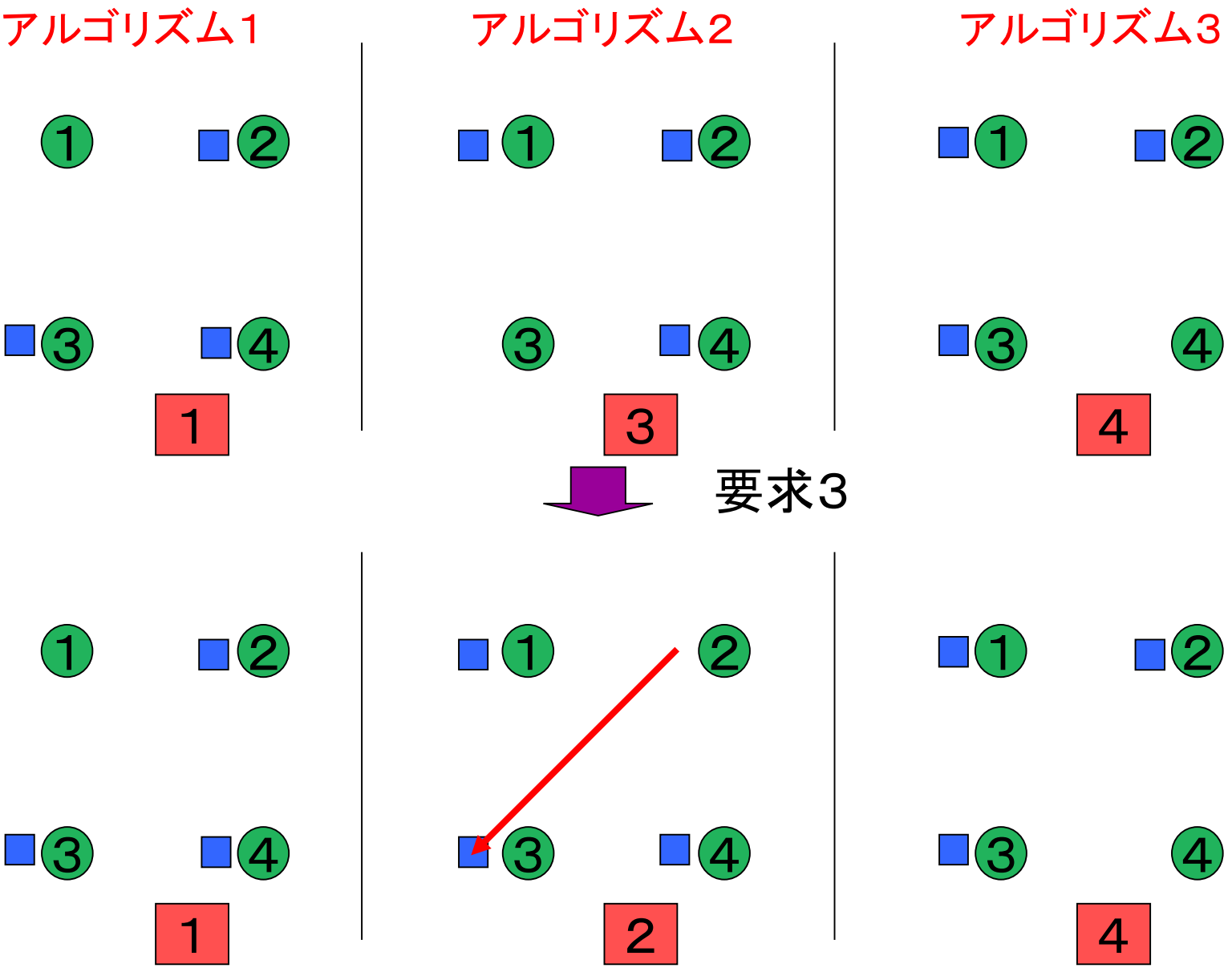
要求：　2　4　1　2　3



要求：　2　4　1　2　3



要求：　２　４　１　２　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。

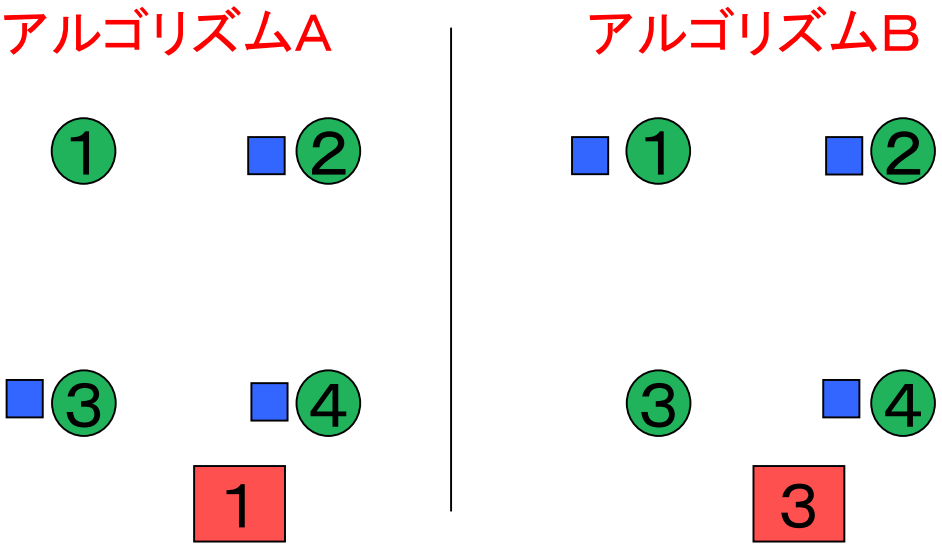
（**それの証明**）

初期状態において違うので、第1リクエスト後も違う。

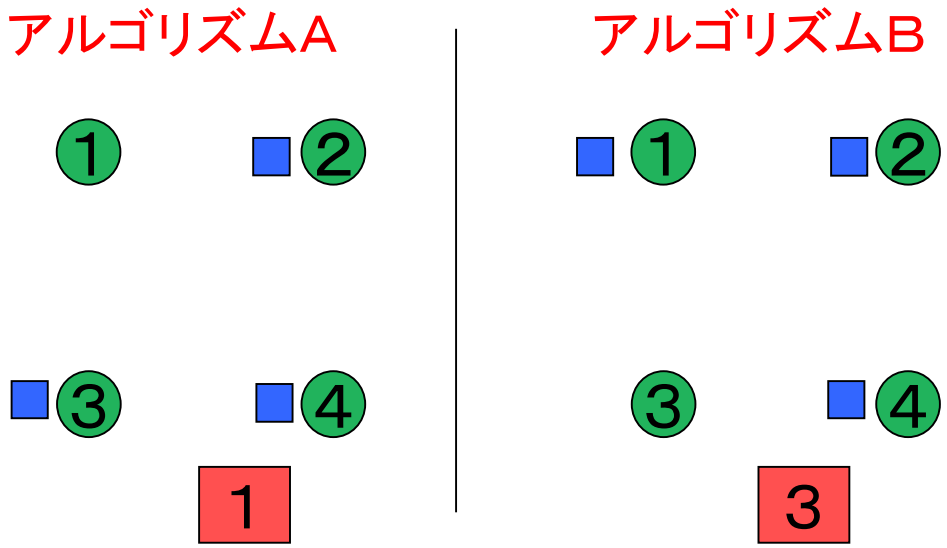
（初期状態において、第1リクエストの来るところには、必ずサーバがあるから。第1リクエスト後は初期状態と同じ。）

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

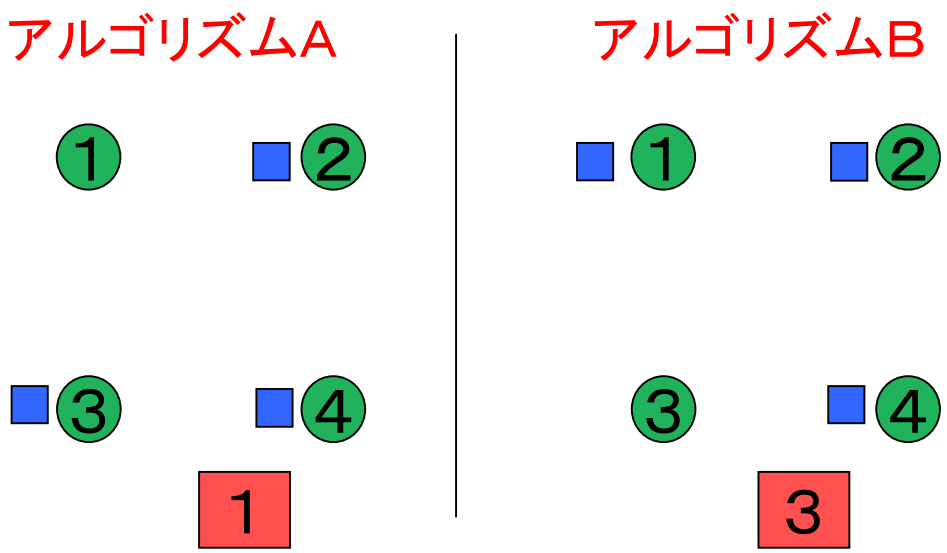
例えば、第 i リクエスト後が以下のようになっていたとしよう。
次のリクエストがどこに来るかで場合分けする。



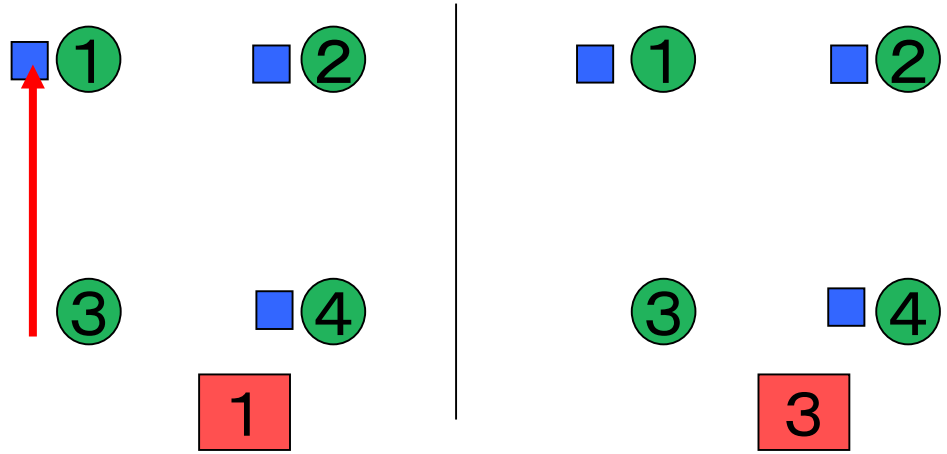
（1）AもBもサーバのある要求点（例えば上記の2）に来る。
AもBも動かさない。
→ i+1 回目のリクエスト後も、状態は異なる。

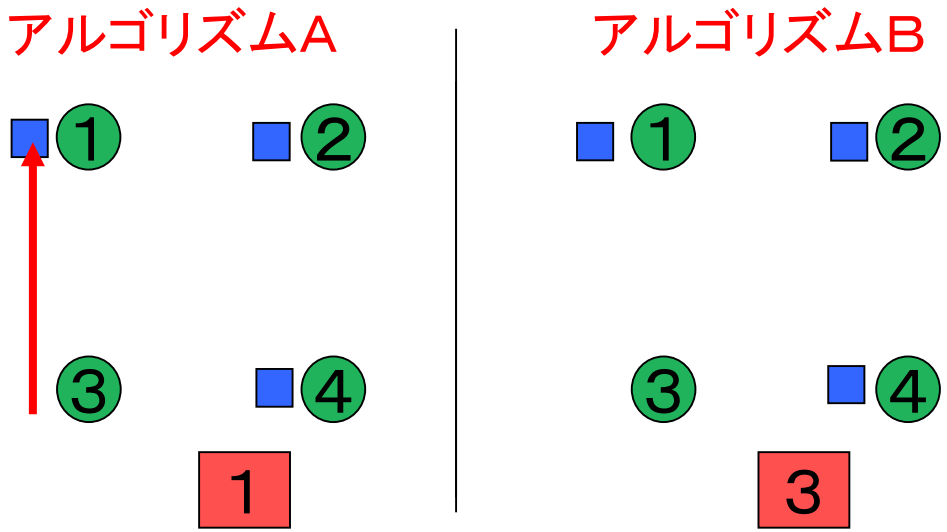


（2）AもBもサーバのない要求点に来る。
　　そんな要求点はない。



（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。



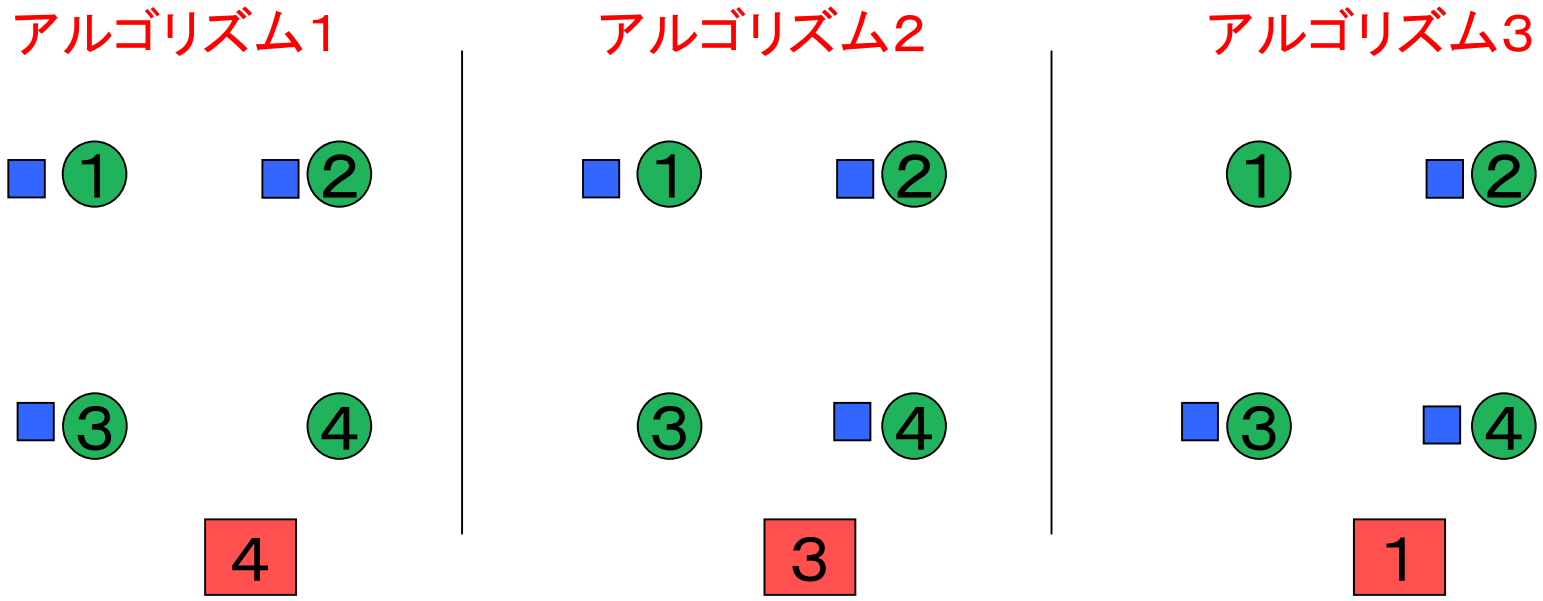


しかし、それなら、ルールより、第 i リクエストの要求は3だった。
しかし、Bは第 i リクエスト後には要求点3にサーバを置いていない。
→ 矛盾

（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



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:

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

↓

少なくとも1つは、コストの総和が

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

以下。OPTは当然それ以下。

アルゴリズムのコスト≧

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

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

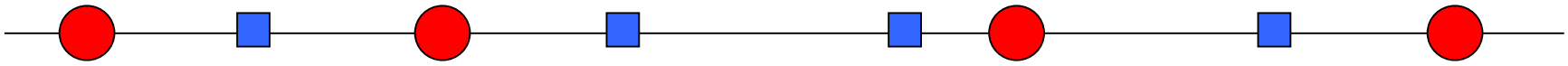
・要求点が$k+1$個しかない場合。

・要求点が$k+2$個しかない場合。

アルゴリズム　Double Coverage (DC)

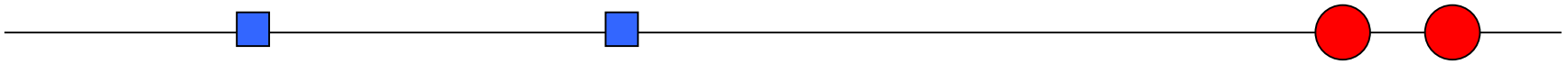
ルール：左端のサーバより左に要求が来たら
→左端のサーバを動かす
右端のサーバより右に要求が来たら
→右端のサーバを動かす
２つのサーバの間に要求が来たら
→それら2つのサーバを同じ距離だけ動かす

k＝4の例

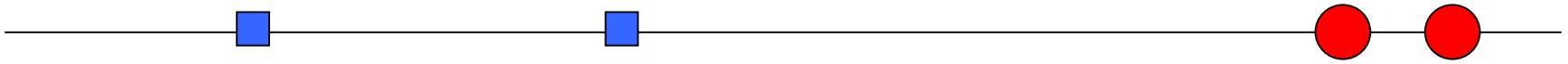


定理：DCの競合比は k 以下である。

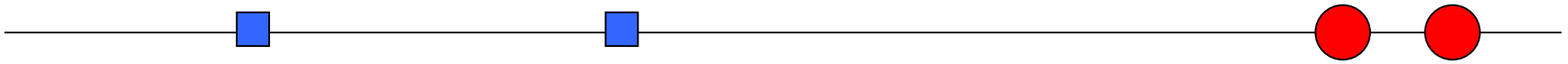
貪欲アルゴリズムの競合比は、ライン上でも無限大



OPT



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　ならしコスト≦1.5×OPTのコスト

・ならし分の合計≧0

→ アルゴリズムのコストの合計≦ならしコストの合計

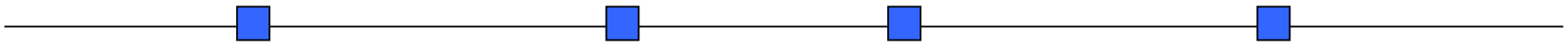
これから、「アルゴリズムの競合比≦1.5」が言える。

定理：DCの競合比は k 以下である。

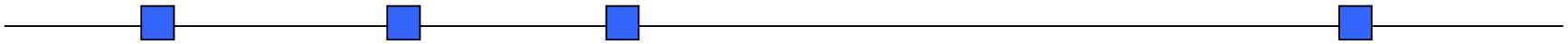
（証明）

ある時点において、

DCの配置



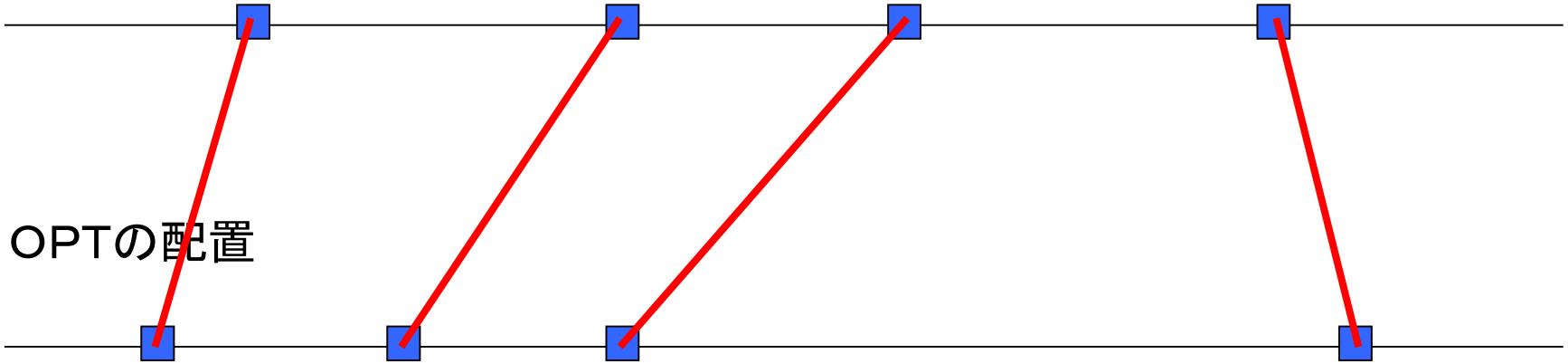
OPTの配置



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。

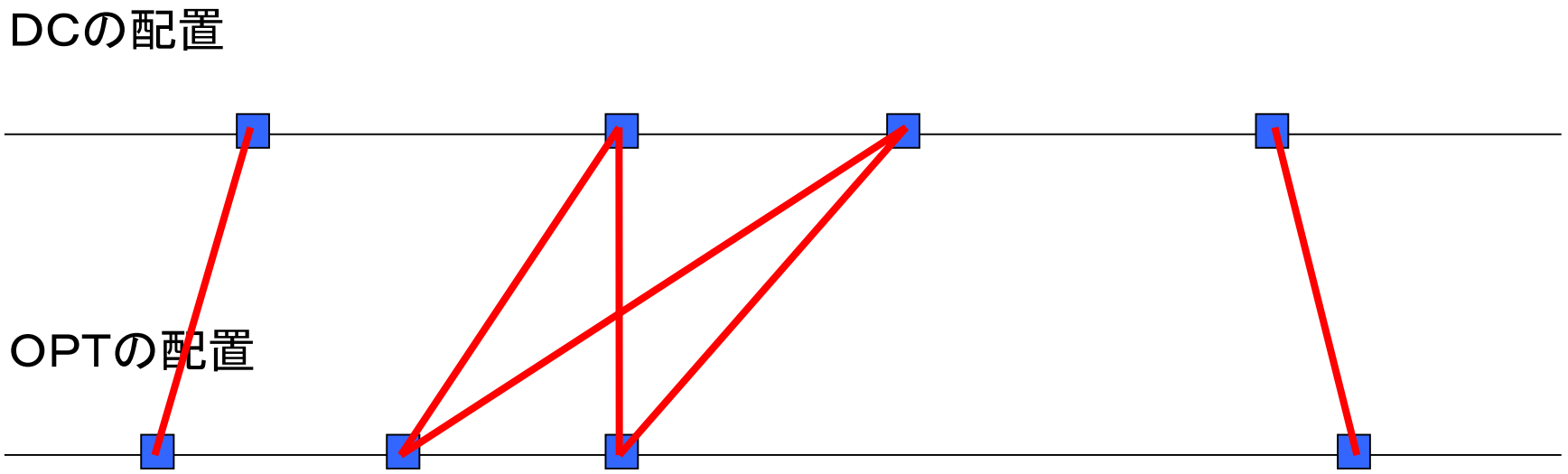
DCの配置



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。

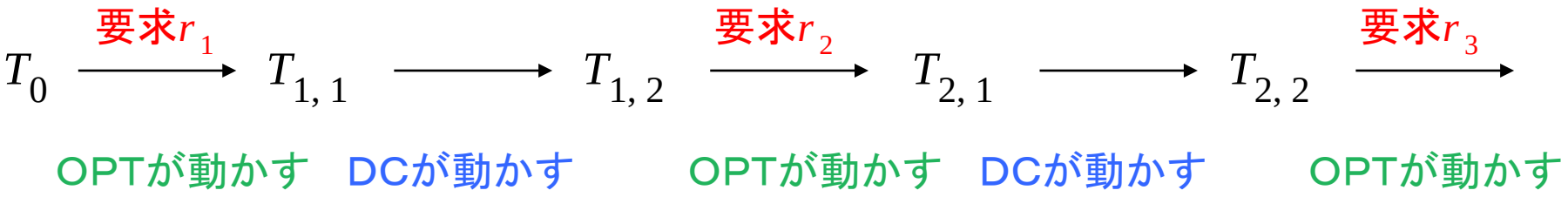


ある時点でのポテンシャル関数（OPTとDCの配置のみから決まる値）を
$T = kM + \Sigma$ とする。

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

ポテンシャル関数の変化を解析する。

$T = kM + \Sigma$

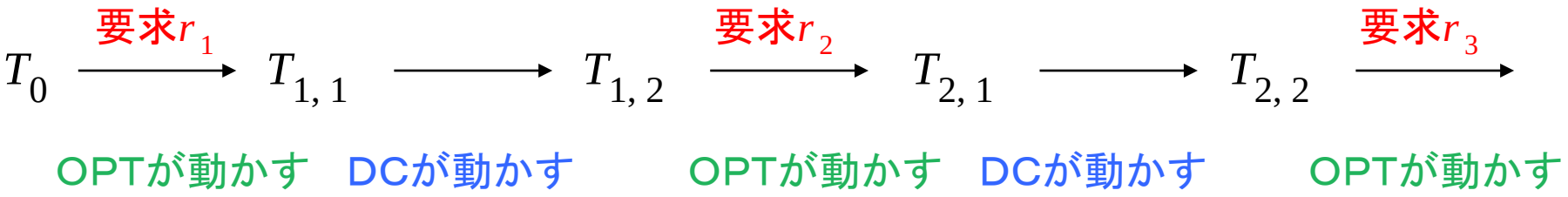


(i) $T_{i,1} - T_{(i-1),2} \leqq kd_i$　　d_i はこの要求に対するOPTのコスト

OPTが動かす番なので、Σ 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。

$$T = kM + \Sigma$$



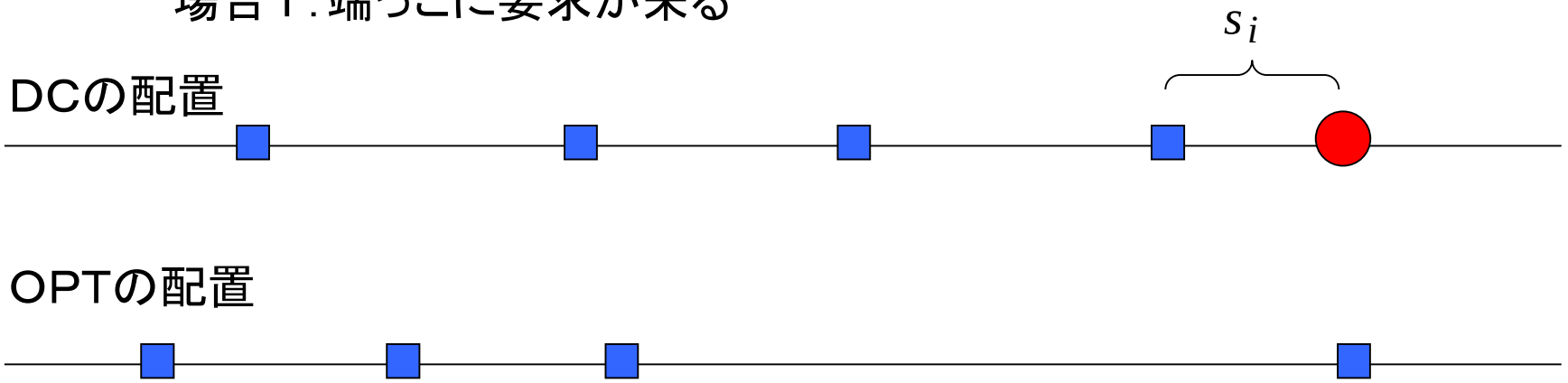
(ii) $T_{i,2} - T_{i,1} \leqq -s_i$

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) $T_{i,2} - T_{i,1} \leqq -s_i$　　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）

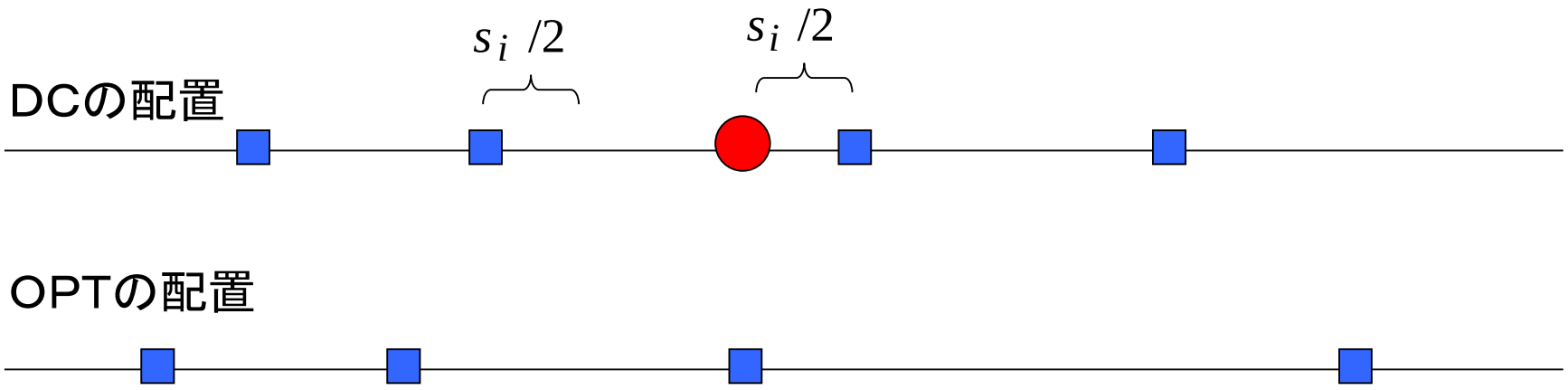
- M はs_i 減る （DCの動かしたサーバは、一番右だった。
  OPTは、少なくとも要求点にはサーバがいる。もしくは、それより
  右にもいるかもしれない。今動かしたDCのサーバは、
  とにかく要求点より右の（OPTの）サーバとマッチしているはず。）
- Σ は$(k\text{-}1)s_i$ 増加する。（自分以外のk-1個のサーバとの距離
  が、それぞれs_i ずつ増えるから。）

↓

全体としてs_i 減る。

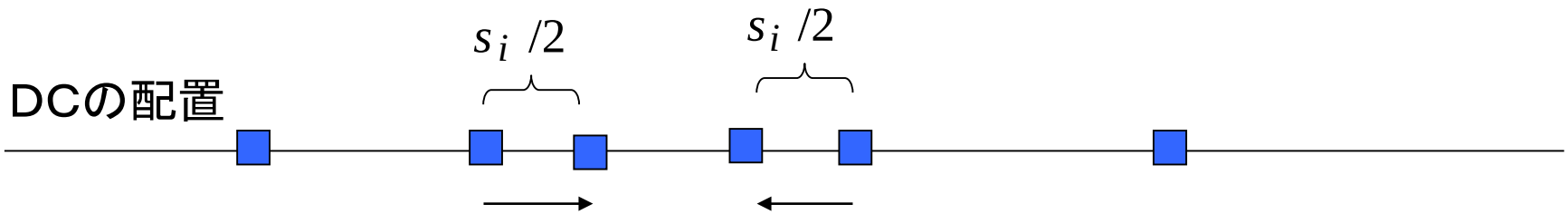
(ii) $T_{i,2} - T_{i,1} \leqq -s_i$　　s_i はこの要求に対するDCのコスト

場合２：間に要求が来る



(ii) $T_{i,2} - T_{i,1} \leqq -s_i$　　s_i はこの要求に対するDCのコスト

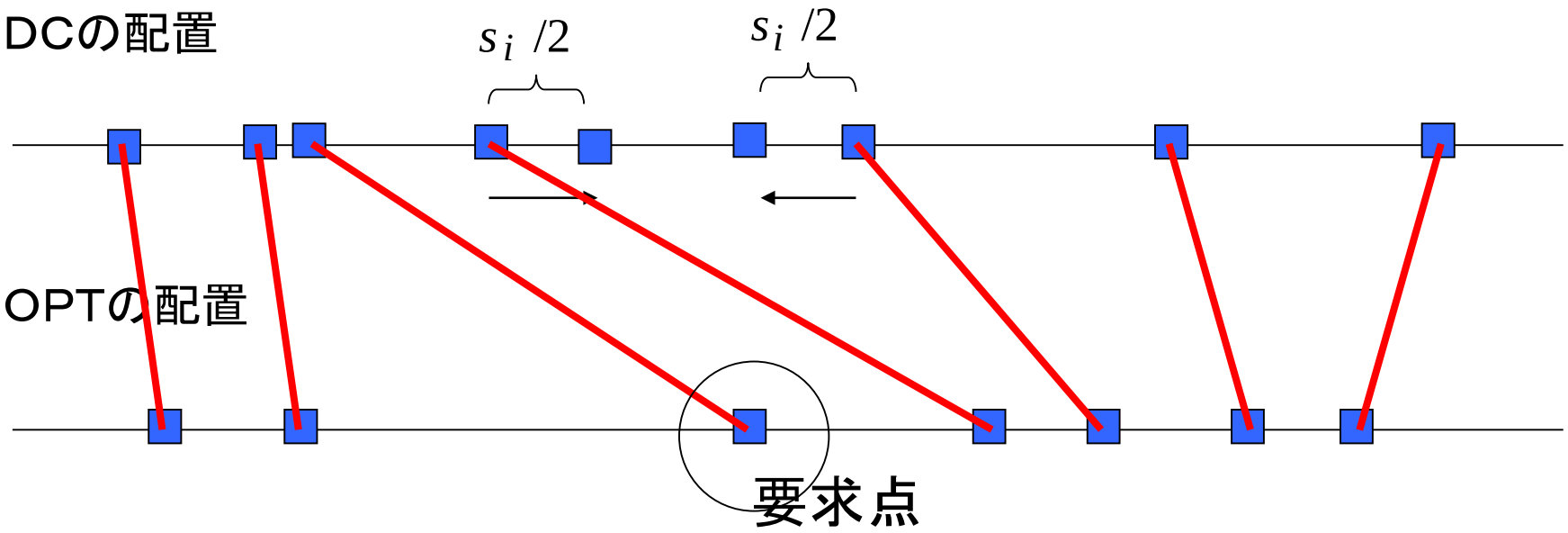
場合２：間に要求が来る



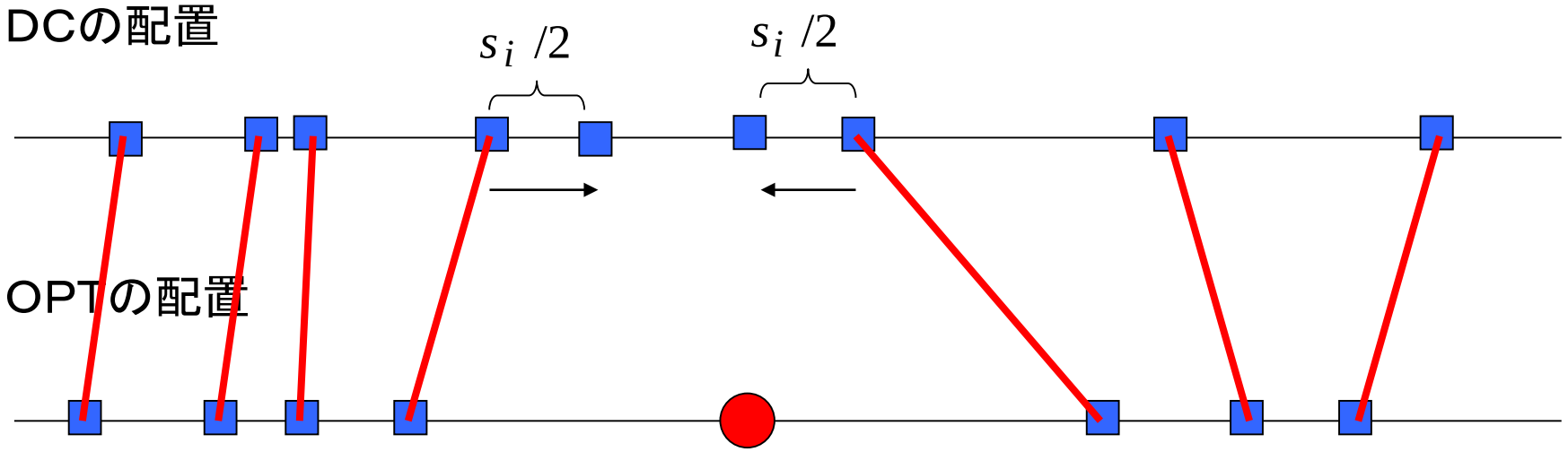
・Σ は、動いた2つのサーバ間はs_i 減る。
それ以外は増えも減りもしない。
（例えば、左にあるサーバだと、１個が s_i /2 遠のいて、もう一個がs_i /2
近づく。つまり、それ対他のサーバの距離の総和は変わらない。）

・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。



・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。



逆に、両方とも増えるとしたら、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) $T_{i,1} - T_{(i-1),2} \leqq kd_i$

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

d_i はこの要求に対するOPTのコスト

s_i はこの要求に対するDCのコスト

(i)と(ii)を足すと

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

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

ならし分

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

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

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

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

$T_{n,2} - T_0$ はnによらない定数。

本当は「ならし分の合計$\geqq 0$」
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、

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

↓

$$\underbrace{s_1 + s_2 + \cdots + s_n}_{\text{アルゴリズムの総コスト}} \leqq \underbrace{k(d_1 + d_2 + \cdots + d_n)}_{\text{OPTの総コスト}} + T_0 - T_{n,2}$$