修士論文

パイプパズルに関する研究

1610095 白山 卓夢

主指導教員　上原 隆平
審査委員主査　上原 隆平
審査委員　池田 心
白井 清昭
平石 邦彦

北陸先端科学技術大学院大学

先端科学技術研究科 [情報科学]

平成30年2月

概要

ゲームとパズルの計算量的な困難さとアルゴリズムの研究は，歴史のある分野で，計算量クラスの特徴づけやアルゴリズムの技法の開発に多くの貢献がある．

タイリングパズルは古典的なパズルの中で大きなカテゴリを形成している．タイリングパズルはタイリングのルールに従って，数多くのバリエーションがある．タイリングパズルの代表例であり，よく知られているパズルにジグソーパズルがある．そして，タイリングパズルの変形として，エッジマッチングパズルが知られている．エッジマッチングパズルでは，それぞれのタイルの辺に色または絵柄が付いている．エッジマッチングパズルのゴールは，同じ色同士で（または，絵柄がつながるように）辺を共有させて，目的の大きさの長方形にすべてのカードを収めた状態である．本論文では，エッジマッチングパズルの新たな種類について研究する．このパズルをパイプパズルと呼ぶ．与えられるカードには，パイプの絵が描かれている．カードの辺にはパイプのいくつかの継ぎ目が現れる．パイプパズルでは，カードを置くとき，パイプが途切れないように辺を共有させなければならない．パイプパズルの入力と目的は，先行して研究されたタイリングパズルと異なる．パイプパズルではカードの他に，パズルを始める前に置かれている 2 つの端点が与えられる．パイプパズルの目的は，すべてのカードを用いて水が漏れないように 2 つの端点の間をつなげることである．

ジグソーパズルとエッジマッチングパズルには，先行研究が存在する．しかし，パイプパズルについては研究がされていない．

本研究ではまず一般化パイプパズルを定式化する．一般化パイプパズルの入力は，正方形のカード，盤，そして盤に設置された 2 つの端点である．盤は長方形とし，盤はカードを配置するセルから構成される．与えられる正方形のカードは，パイプカードとブランクカードの 2 種類がある．パイプカードはパイプの絵が描かれたカードで，ブランクカードは何も描かれていないカードである．パイプカードの辺にはパイプの継ぎ目が現れる．継ぎ目には型があり，継ぎ目は同じ型同士で連結しなければならない．2 つの端点は盤の辺に設置される．具体的には，1 つの端点は盤の左辺，もう 1 つの端点は盤の右辺に設置される．端点はパイプの継ぎ目と同じく型をもつ．

カードを配置するとき，次に示す規則を満たさなければならない：カードを配置して共有する辺が継ぎ目なら，その型と同じ型の継ぎ目を向けなければならない．そして，共有する辺が継ぎ目でないなら，継ぎ目を向けてはならない．この規則を配置規則と呼ぶ．盤上にすべてのカードを配置し，すべてのカードが配置規則に従うとき，この配置を有効な配置と呼ぶ．配置規則に従い，2 つの端点をパイプによってひとつなぎにできたとき，その配置上のパイプカードを順にたどったものを経路という．有効な配置が，すべてのパイプカードを用いた経路をもつとき，その配置をパイプパズルの解と呼ぶ．

この一般化パイプパズルが NP 完全問題であることを証明する．一般化パイプパズルは明らかに NP に属するため，パイプパズルの NP 困難性を証明すればよい．証明には，3-partition 問題

から一般化パイプパズルへの多項式時間還元を用いる．3-partition 問題では，$3m$ 個の正整数が入力として与えられる．この与えられた正整数から 3 つ組集合を m 個作る．集合ごとの 3 つの要素の合計がすべて同じ値になるとき，この集合族を 3-partition 問題の解という．3-partition 問題の入力から，パイプパズルの盤，端点，カードを構成する．そして，3-partition 問題が解をもつ必要十分条件は，還元で構成したパイプパズルが解をもつことであることを示す．

次に，制限したパイプパズルについて多項式時間で解けることを示す．本研究で取り上げる，制限したパイプパズルは 2 つである．1 つ目は高さを定数とするパイプパズルで，2 つ目は高さに制限がなく，パイプカードは有限で，ブランクカードが無限に使えるパイプパズルである．

まず，高さを定数とするパイプパズルが多項式時間・多項式領域で解けることを証明する．そのために，それを解くアルゴリズムを示す．このアルゴリズムは動的計画法に基づく．アルゴリズムではカードの配置に順序を与え，カードの配置した枚数によって次に配置するセルがわかるようにする．すると，カードの配置された領域とそうでない領域を二分する線分ができる．この線分を前線と呼ぶ．このアルゴリズムでは配置したカードの情報をすべては覚えず，配置状態を用いて配置したカードの情報を覚える．配置状態はカードの配置した枚数と前線の上に現れる継ぎ目の連結の状態から構成される．アルゴリズムでは，正当で，解にたどりつける配置状態のみを記憶しパイプパズルを解く．その結果，多項式時間・多項式領域でこのパイプパズルが解けることを示す．

次に，高さに制限がないパイプパズルが，特定の入力を与えられたときに，線形時間で解けることを示す．本研究で取り上げる特定の入力は 2 つである．1 つ目は，幅が 1 であるときのパイプパズルである．2 つ目は，幅が 6 以上の偶数かつ，2 つの端点が設置される行が同じであるときのパイプパズルである．

目 次

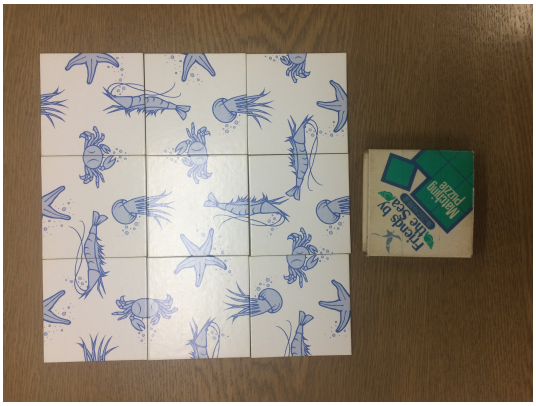
第1章　序論

ゲームとパズルの計算量的な困難さとアルゴリズムの研究は，歴史のある分野で，計算量クラスの特徴づけやアルゴリズムの技法の開発に多くの貢献がある [?]．ゲームとパズルの解き手の立場となれば，ゲームとパズルの計算量を明らかにすることは，そのゲームとパズル自体がどれだけ面白いものかを示す指標となるかもしれない．例えば，あるパズルが計算量的に簡単であるなら，解き手の立場からすると，張り合いがない．一方で，あるパズルが計算量的に困難なら，解くことに張り合いが出るため，解き手は面白いと感じる．このように，ゲームとパズルの計算量を研究することは重要である．

タイリングパズルは古典的なパズルの中で大きなカテゴリを形成している．そして，タイリングパズルはタイリングのルールに従って，数多くのバリエーションがある．タイリングパズルの代表例であり，よく知られているパズルにジグソーパズルがある．ジグソーパズルはピースの集合が与えられる．このピースは多くの場合正方形であり，ピースのそれぞれの辺には，凹の切り込みまたは凸の突起がある．それぞれの辺には凹と凸に従い完全にかみ合うペアがある．かみ合うペアを隣り合わせながらピースを配置する．ジグソーパズルのゴールは，すべてのピースを目的の長方形の形状に配置することである．ただし，目的の長方形の形状は明示されていない．実際のジグソーパズルでは，かみ合うペアが一意的である．つまり，それぞれ辺は一意的なかみ合うペアをもつ．そのため，ジグソーパズルは一意的な解と一意的な配置形状をもつ．しかしながら，計算量的困難さの研究では，ジグソーパズルが，それぞれの辺が 2 つ以上の辺とかみ合うように一般化されている．そして一般化したジグソーパズルでは，計算量的な困難さが劇的に変化する．タイリングパズルの別の一種として，エッジマッチングパズルが知られている．これはジグソーパズルと同じくらい有名でかつ古典的であり，市場には，多くの異なるバリエーションのエッジマッチングパズルがある（図??）．実際，あるエッジマッチングパズルには，最初に解いたものに対して，200 万ドルの賞金が付けられていた．エッジマッチングパズルでは，それぞれのタイルの辺に色または絵柄が付いている．辺に色が付いている場合には，同じ色が付けられた辺同士を合わせながらタイルを置く．辺に絵柄がついている場合には，絵柄がつながるように，タイルを置いていく．エッジマッチングパズルにのゴールは，同じ色同士で（または，絵柄がつながるように）辺を共有させて，目的の大きさの長方形にすべてのカードを収めた配置である（図??）．

(a) 様々なバリエーションのエッジマッチングパズル.

(b) エッジマッチングパズルで与えられるタイル.

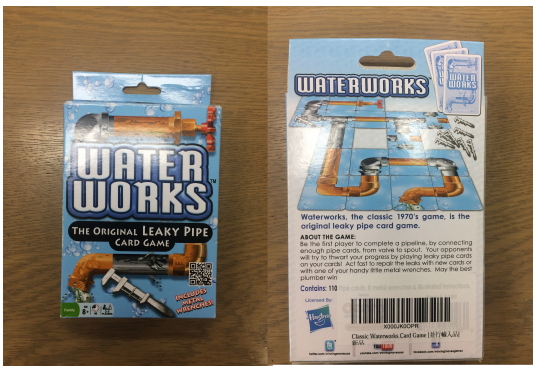


(c) エッジマッチングパズルの解の例.

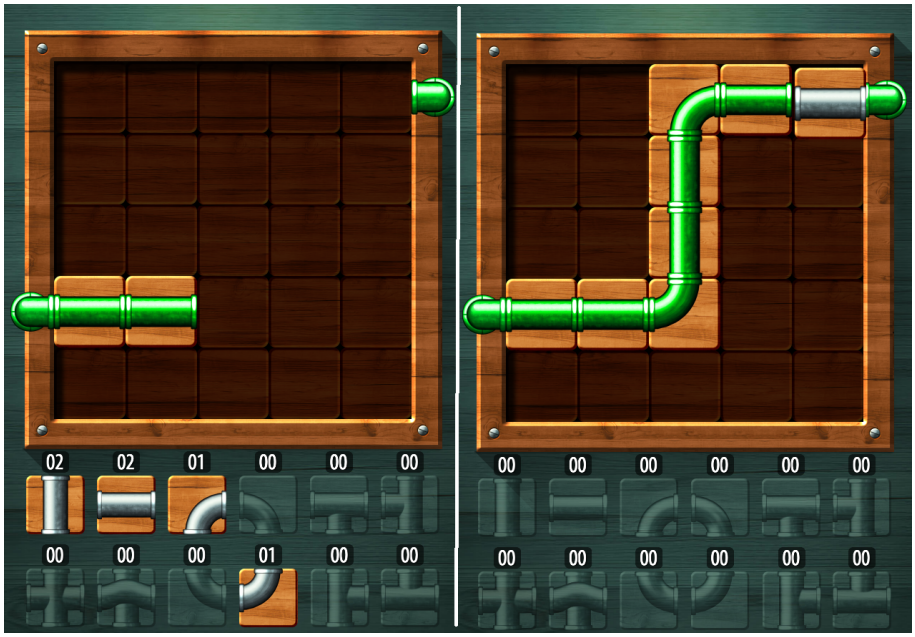
図 1.1: エッジマッチングパズル.

本論文では，エッジマッチングパズルの新たな種類について研究する．このパズルをパイプパズルと呼ぶ．パイプパズルで与えられるカードには，パイプの絵が描かれている．パイプはカードの辺で切られ，カードの辺にはパイプのいくつかの継ぎ目が現れる．パイプパズルでは，カードを置くとき，パイプが途切れないように辺を共有させなければならない．さらに辺を共有させるとき，継ぎ目の位置がちょうど合っている必要がある．パイプパズルのゴールは先行して研究されたタイリングパズルと異なる．パイプパズルでは，カードの他に，パズルを始める前に置かれている端点を 2 つ与える．パイプパズルのゴールは，すべてのカードを用いて，水が漏れないように 2 つの端点の間をつなげることである（図??）.

(a)



(b)



(c) ECO Pawel Jarosz, “ Pipe Puzzle ”

図 1.2: 一般的なパイプパズルの例.

ジグソーパズルとエッジマッチングパズルは，どちらも辺を合わせてピース（タイル）を置くパズルである．そのため，どちらもタイリングパズルに包括される．これらの 2 つのパズルが異なる点は 2 つある．1 つ目は辺の形状である．ジグソーパズルでは辺に凹と凸がある．それに対してエッジマッチングパズルでは辺は直線で，辺の色（または絵柄）を指標にタイルを置いていく（図??）．つまり，エッジマッチングパズルではジグソーパズルに比べて，凹と凸の

ペアがないためタイルを隣接させる自由度が高い．2つ目の異なる点は，ピース（タイル）を収める目的の大きさの長方形が与えられるかそうでないかである．ジグソーパズルでは収める形状を言及されないが，エッジマッチングパズルでは目的の大きさの長方形に収まるようにタイルを配置していく．

パイプパズルはカードの直線な辺を，パイプの絵柄に合わせて置くパズルである．よって，パイプパズルはタイリングパズルに包括され，エッジマッチングパズルと似ているパズルに見える．しかし，パイプパズルの目的は，エッジマッチングパズルとは異なり，辺のパイプの絵柄に合わせてカードを置くだけではなく，2つ端点の間をすべてのカードを用いてパイプでつなぐことである．さらに，すべてのパイプを端点をつなぐために使わなければならないという条件がある．そのため孤立した閉路を作らないようにする必要がある．閉路とは，いずれの端点ともひとつなぎとなっていないパイプである．閉路を作らないようにするということは，パイプパズルでは，単にパイプが途切れないように辺の絵柄を共有させるだけでなく，カードを配置したときの全体像を確認しなければならない．つまり，パイプパズルはエッジマッチングパズルとは種類が異なり，ジグソーパズルとも明らかに異なる．

一般化されたジグソーパズルとエッジマッチングパズルは，どちらもNP-完全問題である[?]．さらに，エッジマッチングパズルは，他のバリエーションについても研究がされている．例えば，$1 \times n$のエッジマッチングパズルである．$1 \times n$のエッジマッチングパズルとは，タイルを収める目的の長方形の大きさが1行のみとなり，タイルを横へ並べて隣接させるエッジマッチングパズルである．このエッジマッチングパズルもNP完全問題である[?]．

このように，ジグソーパズルとエッジマッチングパズルには先行研究が存在する．しかし，パイプパズルについては研究がされていない．

本論文の目的は，パイプパズルの計算量的な困難さを明らかにすることである．そのためにまず，一般化パイプパズル問題を定式化する．そして，一般化パイプパズル問題について計算量的な困難さを明らかにする．さらに，制限を与えたパイプパズルを定式化し，そのパイプパズルが多項式時間で解けることを示す．

第2章　一般化パイプパズル問題の定式化

本章では，一般化パイプパズルを定式化する．

一般化パイプパズル問題の入力は，正方形カードの集合，カードを置く盤，そして盤上の2つの端点の位置である．

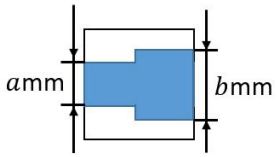
本研究ではカードを正方形にすることで，より一般性を持ったパイプパズルにする．正方形のカードを回転させることで，1枚のカードに4種類の置き方を与えられる．パイプパズルで用いる正方形のカードを大きく2つにわける．1つはパイプの絵が描かれたカード，もう1つは何も描かれていないカードである．それぞれを**パイプカード**，**ブランクカード**と呼ぶ．

本研究で用いるパイプカードのパイプの種類は，分岐のないパイプのみとする．そのときパイプカードの辺には，パイプの継ぎ目がある辺とそうでない辺の2種類がある．ここではパイプの継ぎ目を単に**継ぎ目**という．継ぎ目のある辺と継ぎ目のある辺を合わせることをパイプの連結に見立てている．

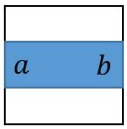
分岐のないパイプのみとすると，パイプの種類は2つとなる．それは直線のパイプと，90° 曲がったパイプである．それぞれのパイプの種類をもつカードを，**直線パイプカード**と**直角パイプカード**と呼ぶ．それぞれのパイプカードに**正位置**を定義する．直線パイプカードの正位置は，上と下に継ぎ目を向けている位置で，直角パイプカードの正位置は，上と右に継ぎ目を向けている位置である．

継ぎ目には**型**があり，継ぎ目は同じ型同士でのみ連結できるものとする．本研究での継ぎ目の型とはパイプの経の太さをモデル化したものと考える（図**??**）．そして継ぎ目はカードの辺の中央に位置している．そのため，カードを回転させても同じ型同士のみで連結できる．$j_1, j_2, \ldots, j_k$ は継ぎ目の型を示す．なお本研究では，継ぎ目の型のモデル化に，辺に現れる継ぎ目の位置を用いない（図**??**）．

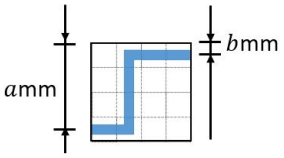
$P = (p_0, p_1, p_2, \ldots, p_n)$ を与えられたカードの種類を表すラベルとする．カードの種類とは，パイプの種類（またはブランク）と継ぎ目の型によって分けた種類である．p_0 はブランクカードを示す．そして $p_1, p_2, \ldots, p_n$ は異なるパイプカードを示す．各パイプカード p_i を $p_i = (X, (j, j'))$ と表す $(1 \leq i \leq n)$．X の値は I または L をとる．I, L はそれぞれ直線パイプカード，直角パイプカードであることを示す．(j, j') はカードを正位置としたときの継ぎ目の型を示す．p_i の $X = I$ であるとき，j は上の継ぎ目の型を示し，j' は下の継ぎ目の型を示す．p_i の $X = L$ であるとき，j は上の継ぎ目の型を示し，j' は右の継ぎ目の型を示す．j, j' はそれぞれ $j_1, j_2, \ldots, j_k$ のいずれかの値をとるものとする．つまり型は k 種類ある．



(a) 継ぎ目の径が異なるパイプ



(b) モデル化したパイプの表現



(c) 本研究では扱わない 継ぎ目の位置が異なるパイプ

図 2.1: カードのモデル化．本研究では，継ぎ目の径の大きさを継ぎ目の型としてモデル化する．継ぎ目の位置が異なるパイプは本研究では扱わない．

カードは**盤**の中へ重ならないように配置する．盤は高さ h，幅 w の長方形 $w \times h$ とする．ただし h, w は正の整数である．この長方形は $N(= wh)$ 個のセルからなる．このセルへカードを配置していく．それぞれのセルに座標を与える．(i, j) を i 列 j 行目のセルとする．

パズルで使うカードの枚数は入力で与えられる．与えられるカードの枚数は合計 $N = hw$ 枚とする．**デッキ** $D = (d_0, d_1, d_2, \ldots, d_n)$ を与えられるカード枚数とする．d_i はそれぞれ p_i が入力で与えられる枚数を示す．ただし，$d_0 + d_1 + d_2 + \cdots + d_n = N$ とする．

盤の縁には**端点**を 2 つ設置する．端点には継ぎ目を接続する．それぞれの端点は盤の右辺と左辺へそれぞれ設置されるとする．端点はパイプの継ぎ目と同じく型をもつ．端点の設置位置と型は入力として与える．それぞれの端点を s と t で表し，s は盤の左辺，t は盤の右辺に設置されるものとする．

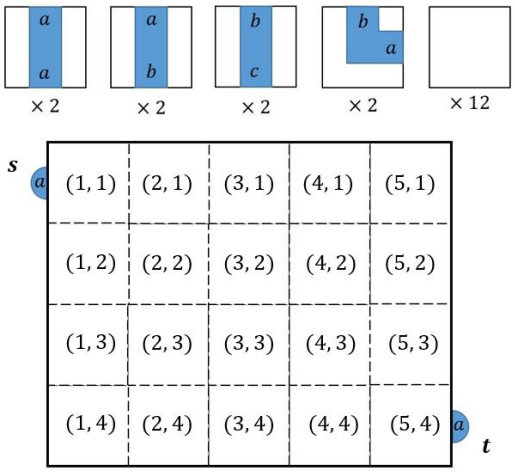


図 2.2: 一般化パイプパズルの直感的な表現．このときのカードの青い部分がパイプを表し，そこに書かれた記号が継ぎ目の型を表す．盤の大きさは 5×4 であり，それぞれのセルの座標を図内に示す．左，右の青い半円が端点であり，それぞれ s, t である．

一般化パイプパズル問題とは，与えられた盤に与えられたカードを矛盾なく敷き詰められるかどうかを問う問題である．以下で矛盾のない敷き詰めを定義する．i 列 j 行目のセルに配置されたカード（または端点）を，$\mathrm{card}(i,j)$ で表現する．i,j の値はそれぞれ $0 \leq i \leq w+1, 0 \leq j \leq h+1$ の範囲をとる．$1 \leq i \leq w$ かつ $1 \leq j \leq h$ なら盤内を示し，i,j がそれ以外の値なら盤外を示す．具体的には，次のように定義できる．

$$\mathrm{card}(i,j) = \begin{cases} s\,(\text{始点}) & (i=0, 1 \leq j \leq h) \\ t\,(\text{終点}) & (i=w+1, 1 \leq j \leq h) \\ p_0\,(\text{ブランクカードが配置されているとき，} \\ \quad \text{または盤外の座標の端点以外を参照しているとき}) & \\ p_m\,(\text{パイプカード } p_m\,(m=1,2,\ldots,n) \text{ が配置されているとき}) & \\ 0\,(\text{まだカードが配置されていないとき}) & (1 \leq i \leq w, 1 \leq j \leq h) \end{cases}$$

端点 s,t はそれぞれ盤の左辺，右辺に設置されている．そのため，s の位置は $i=0, 1 \leq j \leq h$ のいずれかの位置であり，t の位置も $i=w+1, 1 \leq j \leq h$ のいずれかである．例えば，図**??**に示す位置に s,t があるなら，$\mathrm{card}(0,1)=s, \mathrm{card}(6,4)=t$ である．端点が設置されていない盤外は，継ぎ目を向けてはいけない領域である．したがって，$\mathrm{card}(i,j)$ はこのとき，ブランクカードが配置されていることと定義した．$\mathrm{card}(i,j)$ がとる値の例を示す．図**??**に示す位置 $(2,2)$ にカード p_1 が配置されているとする．このとき，$\mathrm{card}(2,2)=p_1$ である．

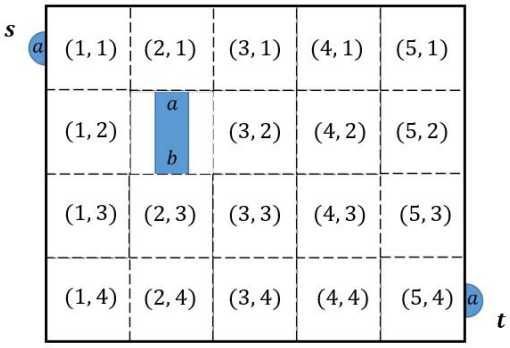


図 2.3: カードを配置している盤面の例

セルを調べて，カードが配置されているセルであるなら，そのカードの継ぎ目の向きと型を確認する必要がある．継ぎ目の向きと型を確認することで，セルが共有する辺の状態を得ることができる．i 列 j 行目に置かれたカード $\mathrm{card}(i,j)$ の向きと型を $\mathrm{dir}(i,j)$ で表現する．具体的

には次に定義する．

$$\mathrm{dir}(i,j)=\begin{cases}\emptyset & (\mathrm{card}(i,j)=p_0\text{のとき})\\ \{C_{p_m},C'_{p_m}\} & (\mathrm{card}(i,j)=p_m\,(m=1,2,\ldots,n)\text{ のとき})\\ \{C_s\} & (\mathrm{card}(i,j)=s\text{ のとき})\\ \{C_t\} & (\mathrm{card}(i,j)=t\text{ のとき})\\ 0 & (\mathrm{card(i,\ j)}{=}0\text{ のとき})\end{cases}\qquad(0\le i\le w, 0\le j\le h)$$

(i,j) にカードが配置されているときに取りうる値を，表**??**に示す．ただし，$\hat{T}=\{T_{j_1},T_{j_2},\ldots,T_{j_k}\}$,

表 2.1: 配置されたカード p_m がとりうる値．

$\mathrm{card}(i,j)=p_m$	正位置	正位置から 90° 右に回転	正位置から 90° 左に回転	正位置から 180° 回転
p_m が直線パイプカード	$C_{p_m}\in\hat{T}$ $C'_{p_m}\in\hat{B}$	$C_{p_m}\in\hat{R}$ $C'_{p_m}\in\hat{L}$	$C'_{p_m}\in\hat{T}$ $C_{p_m}\in\hat{B}$	$C'_{p_m}\in\hat{R}$ $C_{p_m}\in\hat{L}$
p_m が直角パイプカード	$C_{p_m}\in\hat{T}$ $C'_{p_m}\in\hat{R}$	$C_{p_m}\in\hat{R}$ $C'_{p_m}\in\hat{B}$	$C_{p_m}\in\hat{B}$ $C'_{p_m}\in\hat{L}$	$C_{p_m}\in\hat{L}$ $C'_{p_m}\in\hat{T}$

$\hat{B}=\{B_{j_1},B_{j_2},\ldots,B_{j_k}\}$, $\hat{R}=\{R_{j_1},R_{j_2},\ldots,R_{j_k}\}$, $\hat{L}=\{L_{j_1},L_{j_2},\ldots,L_{j_k}\}$とする．集合$\hat{T},\hat{B},\hat{R},\hat{L}$はそれぞれ，配置されているカードがもつ継ぎ目の向きを表す集合である．$\hat{T},\hat{B},\hat{R},\hat{L}$ はそれぞれ，カードの上，下，右，左の向きに継ぎ目をもつことを示す．そして，それぞれの要素の添え字は，その方向を向く継ぎ目の型である．例として，(i,j) が図**??**のような継ぎ目と型なら，$\mathrm{dir}(i,j)=\{T_{j_1},B_{j_2}\}$ とする．C_s,C_t はそれぞれ端点 s,t がもつ継ぎ目の向きとその型を示

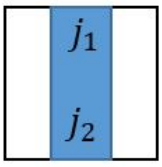


図 2.4: セルへ配置したカードの継ぎ目方向とその型．

す．つまり，$C_s\in\hat{L},C_t\in\hat{R}$ である．

それらの継ぎ目と型に従い，カードを配置して型が矛盾しないようにしなければならない．この規則を**配置規則**と呼ぶ：

(i,j) へカードを配置するとき，上下左右のセルのカード $\mathrm{card}(i,j-1),\mathrm{card}(i,j+1),\mathrm{card}(i+1,j),\mathrm{card}(i-1,j)$ を確認する．card() が 0 なら，その方向に向ける辺

は，継ぎ目でも空白でもよい．card() が 0 でないなら，(i,j) が共有する辺の型の整合性を考慮してカードを隣り合わせなくてはならない．つまり，共有する辺が継ぎ目ならその型と同じ継ぎ目を向けなければならない．例えば，$B_{j_1} \in \mathrm{dir}(i,j-1)$ なら，$T_{j_1} \in \mathrm{dir}(i,j)$ でなくてはならない．そして，共有する辺が継ぎ目でないなら継ぎ目を向けてはならない．例えば，$\mathrm{dir}(i,j-1) \cap \hat{B} = \emptyset$ なら，$\mathrm{dir}(i,j) \cap \hat{T} = \emptyset$ でなくてはならない．

この配置規則に従い盤へカードを配置していく．

定義 1. 盤面上にすべてのカードを配置し，N 枚すべてのカードが配置規則に従うとき，この配置を**有効な配置**と呼ぶ．有効な配置では，継ぎ目はいずれかの継ぎ目と必ず連結する．

定義 2. 配置規則に従いカードを配置できたとする．このとき，ある辺を共有する両側のパイプカードの継ぎ目が合致している．これをパイプの**連結**という．

定義 3. 配置規則に従いカードを配置できたとし，ある 2 つのパイプカード A, B があるとする．A, B の間を，隣接するパイプカードの列でつなぐことができ，その列の中では，カードの重複はなく，隣り合うどの 2 枚のカードもパイプが連結であるとき，A と B は**ひとつなぎ**であるという．

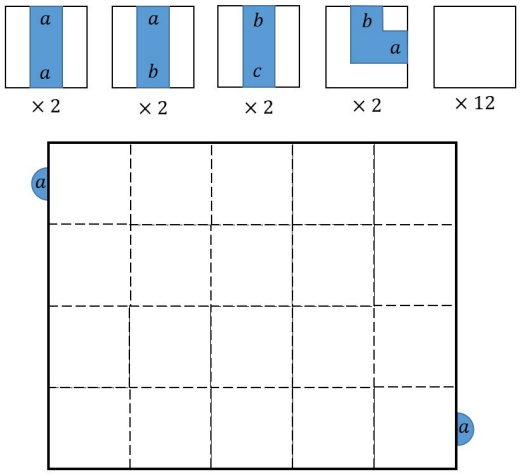
定義 4. 配置規則に従ってカードが配置できたとする．このときさらに，端点 s と t がひとつなぎとなり，s と t をつなぐパイプカードの列にすべてのパイプカードが現れているとする．このとき，そのひとつなぎのカード列を s から t に順にたどって得られるカードの列を s から t への**経路**という。

パイプパズルとは次に示す問題である：

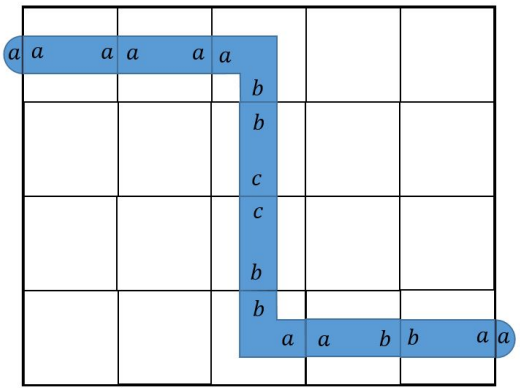
入力:正整数 w と h，カードのラベル集合 $P = (p_0, p_1, p_2, \ldots, p_n)$，デッキ $D = (d_0, d_1, d_2, \ldots, d_n)$（ただし，$d_i$ は正整数で，$\Sigma d_i = wh = N$ とする）, 正整数 j と j'（ただし $\mathrm{card}(0,j) = s, \mathrm{card}(w+1, j') = t$（$1 \leq j, j' \leq h$）とする）.

出力：このパイプパズルは経路をもつか.

経路をもつとき，その有効な配置をパイプパズルの**解**と呼ぶ．入力の例とその解の例を図**??**に示す.



(a) 一般化パイプパズルの例



(b) その解の例

図 2.5: yes-instance である一般化パイプパズルの例.

第3章　一般化パイプパズルの困難さ

本章では，一般化パイプパズルの困難さを示す．

本論文の還元には次に示す 3-partition 問題を用いる：

入力：$3m$ 個の正の整数の多重集合 $A = \{a_1, a_2, \ldots, a_{3m}\}$，ただし，$\Sigma = (a_1 + a_2 + \cdots + a_{3m})/m$ は正の整数で，かつ $\frac{\Sigma}{4} < a_i < \frac{\Sigma}{2}$ とする．

出力：m 個の 3 つ組集合を作り，それぞれ集合の要素の合計を Σ にできるかどうか．

3-partition 問題は NP 完全問題であることが知られている [?].

定理 1. 一般化パイプパズルは NP 完全問題である．

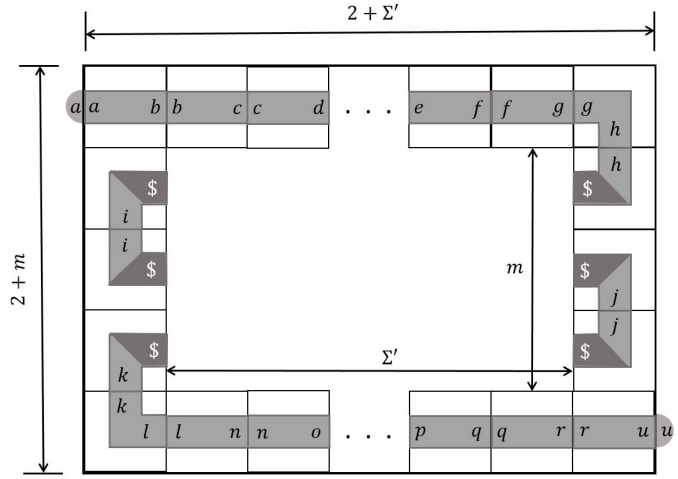
証明：まず一般化パイプパズルは明らかに NP に属するので，NP 完全性を示すには NP 困難性を示せばよい．NP 困難性は 3-partiotion 問題からの多項式時間還元を示すことで証明する．3-partition 問題を一般化パイプパズルへ還元するため，まず盤と端点を構成する．

盤・端点の構成：3-partition 問題のインスタンスから盤と端点を構成する．構成する盤の列数は $\Sigma' + 2$ とする．ただし $\Sigma' = \Sigma + 3m$ とする．盤の行数は，3 つ組の数 m が奇数か偶数かによって変える．m が奇数の場合は盤の行数を $m + 2$ 行とする．それに対して，m が偶数の場合は盤の行数を $m + 3$ 行とする．端点 s は座標 $(0, 1)$ に設置する．そして端点 t は，m が奇数なら座標 $(\Sigma' + 3, m + 2)$，m が偶数なら座標 $(\Sigma' + 3, m + 3)$ へ設置する．s, t の型は，次に示すフレームと合わせて決める．

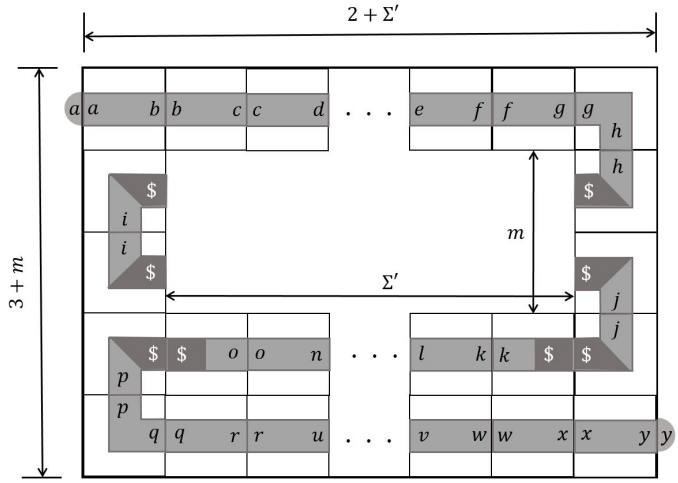
次に N 枚のカードを構成する．カードの構成を 2 つに分けて次に示す．1 つ目は**フレーム**の構成である．フレームとは，m の値により構成されるカードのセットの配置である．2 つ目は**整数カード**の構成である．整数カードとは，A の要素に対応して構成されるカードのセットである．

フレームの構成：盤面の外周部分を図**??**に示したように配置できるカードのセットを，フレームカードという．フレームは図**??**に示すフレームカードの配置とする．フレームの内部には，大きさ $\Sigma' \times m$ の穴があり，穴の中へ整数カードを配置していく．ただし，フレームを構成するとき，フレームカードが共有している継ぎ目には一意的な型を付け，それ以外の継ぎ目とは隣接できないようにする．さらに，端点と隣接する継ぎ目の型にも一意的な型を用いる．フレームの内部の穴に面する継ぎ目は汎用の型 \$ とする．

フレームの輪郭の大きさは，m が奇数なら $(\Sigma'+2)\times(m+2)$ とし，m が偶数なら $(\Sigma'+2)\times(m+3)$ とする．このとき，構成されるカードの枚数は，m が奇数なら $(\Sigma'+2)(m+2)-\Sigma' m$ で，m が偶数なら $(\Sigma'+2)(m+3)-\Sigma' m$ である．



(a) m が奇数のときのフレームの構成方法．



(b) m が偶数のときのフレームの構成方法．

図 3.1: フレームカードによるフレームの構成方法．フレームの配置は一意に決まる．

整数カード：それぞれの正の整数 a_i から整数カードを構成する．正の整数 a_i から $a'_i = a_i + m$ 枚のカードのセットを構成していく．

a'_i 枚のうち 2 枚のカードは，継ぎ目の片方を汎用の型 \$，もう片方を型 i で構成される直線パイプカードとする．この型 i は，整数 a_i ごとの一意的な型とする．2 枚のカードを除いた残りのカードは，継ぎ目の型が両端 i である直線パイプカードとする．i は一意的な型なので，a_i か

ら構成された整数カードは，必ず図**??**に示す $a'_i \times 1$ の長方形状に配置される．

このとき，構成されるカードの枚数は $m\Sigma'$ である．フレームと整数カードのそれぞれで構成されたカードを合わせた枚数をデッキの枚数 N 枚と定める．

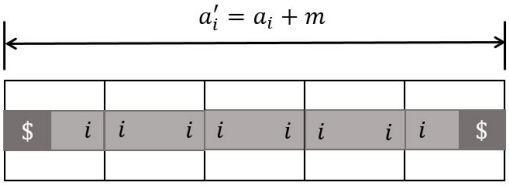


図 3.2: a_i から構成される整数カードと，一意的な配置.

上記の還元は明らかに多項式時間還元である．したがって，元の 3-partition 問題が解をもつ必要十分条件は，還元で構成したパイプパズルが解をもつことであることを示せばよい．

まず，3-partition 問題が解をもつとき，パイプパズルも解をもつことを示す．3-partition 問題 $A = \{a_1, a_2, \ldots, a_{3m}\}$ が解をもつと仮定する．すると，m 個の 3 つ組 $A_1 = \{a_1^1, a_2^1, a_3^1\}, A_2 = \{a_1^2, a_2^2, a_3^2\}, \ldots, A_m = \{a_1^m, a_2^m, a_3^m\}$ はそれぞれ合計が Σ となる．このとき，a_1^i, a_2^i, a_3^i から構成されるそれぞれの整数カードは，型 \$ を共有させて 1 列に連結できる．整数カードを 1 列に連結したとき，長さは $a_1^{i\prime} + a_2^{i\prime} + a_3^{i\prime}$ となる．この長さは穴の 1 行の長さ $\Sigma' = \Sigma + 3m$ と等しくなる．したがって，それぞれの 3 つ組 A_i から構成した整数カードを，穴の i 行目へ敷き詰められる．こうして，すべての整数カードをフレーム内の穴へ敷き詰めることができ，有効な配置を得ることができる．これが s と t をひとつなぎにする経路となることは明らかである．よって，構成したパイプパズルは解をもつ．

次に，構成したパイプパズルが解をもつとき，元の 3-partition 問題も解をもつことを示す．構成したパイプパズルが解をもつと仮定する．すると，フレームカードの配置の一意性より，フレーム内に整数カードが敷き詰められている．そして，各整数カードの枚数は $\frac{\Sigma'}{4}$ より大きく，かつ $\frac{\Sigma'}{2}$ より小さく制限されているため，穴のそれぞれの行には，ちょうど 3 つの整数から構成された整数カードが敷き詰められている．穴の i 行目を敷き詰めた整数カードから，3 つの整数の組 a_1^i, a_2^i, a_3^i が得られる．$a_1^{i\prime} + a_2^{i\prime} + a_3^{i\prime}$ は Σ' となるため，$a_1^i + a_2^i + a_3^i$ は Σ になる．よって，m 個の 3 つ組すべてが合計 Σ になる．つまり元の 3-partition 問題が解をもつ．

よって一般化パイプパズルは NP 完全問題である． □

第4章　多項式時間で解けるパイプパズル

この章ではパイプパズルに制限を与える．そして，制限したパイプパズルについて多項式時間で解けることを示す．まず本章では，継ぎ目の型を1種類のみとする．なお，市販のパイプパズルはこの制限を満たす．この章で取り上げるパイプパズルは次の2つである．

- 高さ h を定数とするパイプパズル．
- 高さ h に制限がなく，パイプカードは有限で，ブランクカードが無限に使えるパイプパズル．

本章のパイプパズルのカードは型が1種類なので，p_0, p_1, p_2 へ明示的にカードの種類を与える．これ以降，p_0, p_1, p_2 はそれぞれ，ブランクカード，直線パイプカード，直角パイプカードと呼ぶ．直線パイプカードの継ぎ目の向きは $\{T, B\}, \{R, L\}$ のいずれかとなる．そして直角パイプカードの継ぎ目の向きは $\{T, R\}, \{R, B\}, \{B, L\}, \{L, T\}$ のいずれかとなる．また，それぞれのカードの枚数はデッキ $D = (d_0, d_1, d_2)$ で与えられるものとする．

4.1　高さが定数のパイプパズル

この節では，盤の高さ h を定数とする．このときパイプパズルは多項式時間・多項式領域で計算可能であることを示す．それを証明するため，具体的にパイプパズルを解くアルゴリズムを示す．このアルゴリズムは動的計画法に基づいている．

本アルゴリズムでは，カードの配置に順序を与える．その順序は次の疑似コードで決まる：

for $i = 1$ から w まで **do**
  for $j = 1$ から h まで **do**
    (i, j) へカードを配置する．
  end for
end for

順序に沿ってカードを配置していくと，盤をふたつの領域に分けることができる．それは，カードを配置した領域とそうでない領域である．カードを配置した領域を**充足領域**と定義する．

盤には充足領域とそうでない領域を二分する線分がある．この線分は複数のセルの辺で構成される．この線分を**前線**と呼ぶ（図**??**）．前線は盤に常に現れ，盤の上辺のある点と下辺のある点をつなぐ最短の線分である．定義より，カードの配置枚数によって，前線は一意に決まる．

前線のそれぞれの線分について，一方（左か上）はカードが置かれており，他方（右か下）はカードが置かれていない．以下，前線の各線分を，置かれている方のカードの辺と同一視する．また，これ以降，解になりうる前線だけを考察の対象とする．

前線を構成する辺は h 本または $h+1$ 本である．それぞれの辺は，継ぎ目である辺かそうでない辺となる．もし継ぎ目である場合，パイプは充足領域内を通り，他の継ぎ目（または端点 s の) とひとつなぎでなければならない．これら前線の辺がもっている状態を**連結ラベル**で表す．$L = \{l_1, l_2, \ldots, l_{h'}\} \cup \{l_s, l_B\}$ を連結ラベル集合とする．それぞれの連結ラベルがもつ意味を次に示す：

- $l_i\,(i = 1, 2, \ldots, h')$ は継ぎ目であり，もう 1 つの l_i とひとつなぎである．
- l_s は端点 s とひとつなぎの継ぎ目である．
- l_B は継ぎ目ではない辺（継ぎ目を向けてはいけない辺）であることを示す．

上記の連結ラベルとカードの枚数により，盤にカードを配置した状態を表現する．盤にカードを配置した状態を，**配置状態**と呼び，$\mathrm{S}(u_0, u_1, u_2, c_1, c_2, \ldots, c_{h+1})$ と書くことにする．ただし，u_0, u_1, u_2 は，それぞれ p_0, p_1, p_2 を盤へ配置した枚数とする．前線を構成する辺が h 本であるとき，$c_i\,(1 \leq i \leq h+1)$ は前線の i 本目の辺の連結を表す状態とする．前線を構成する辺が h 本であるときは，$c_1 = l_B$ と定義し，$c_{i+1}\,(1 \leq i \leq h)$ は前線の i 本目の辺の連結を表す状態とする．つまり $c_i \in L$ である．

多項式時間・領域で問題を解くため，アルゴリズムでは配置したカードのすべての情報は覚えない．そして解にたどりつける配置状態だけを効率よく管理する．そのために配置状態の**妥当性**を導入する．

定義 5. 妥当な配置状態とは，次の 3 つを満たす配置状態である．

- その配置状態は，与えられたカードを配置規則に従って配置して，実現可能である．
- その配置状態は，今 s, t 間に経路が構成されている，またはこれからカードを追加することで s, t 間に経路が構成できる可能性がある．
- その配置状態には閉路になるパイプがない．

妥当な配置状態を用いて，パイプパズルを解くためのアルゴリズムを示す．このアルゴリズムは，配置の順序に従い，カードの使用枚数が少ない順に，妥当な配置状態に限りカードの追加処理をする．追加処理の後，妥当でないと判断できたならその配置を破棄して，そうでなけ

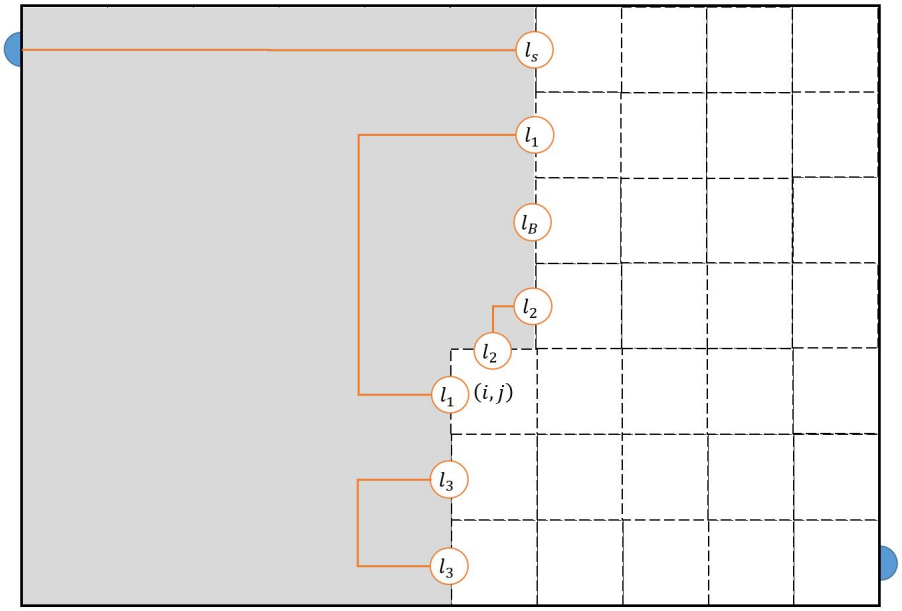


図 4.1: 妥当な配置状態の例．この図は配置状態を直感的に表している．灰色の領域は充足領域を表している．この配置状態は，カード枚数をそれぞれ u_0, u_1, u_2 とすれば，$\mathrm{S}(u_0, u_1, u_2, l_s, l_1, l_B, l_2, l_2, l_1, l_3, l_3)$ となる．この配置状態に次のカードを追加するときは，セル (i, j) へカードを配置する．

れば追加処理をした新たな配置状態を妥当な配置状態とする．これを繰り返すことで，有効な配置のときに妥当な配置状態があれば，このパイプパズルに解があることを示すことができる．そして，妥当な配置がないときには，パイプパズルに解がないことを示すこともできる．

アルゴリズムの本体：アルゴリズムの中で，配置状態を $3 + h + 1$ 次元配列として表し，$\mathrm{S}[u_0, u_1, u_2, c_1, c_2, \ldots, c_{h+1}]$ とする．配列の要素には 0 または 1 が格納される．0 は妥当でない配置状態，1 は妥当な配置状態とする．

カードの使用枚数 $u_0 + u_1 + u_2$ が少ない順に，妥当な配置状態を探す．そして，この妥当な配置状態に対してカードの追加処理をする．

main(){
すべての S[] を 0 で初期化する．
初期盤面 $\mathrm{S}[0, 0, 0, c_{0,1}, \ldots, c_{0,h+1}] = 1$ を妥当とする．ただし，$c_{0,1}, \ldots, c_{0,h+1}$ は盤の左辺の連結ラベル．ある i で $c_{0,i} = l_s$ となり，他はすべて $c_{0,j} = l_B$ となる．i は端点 s の座標として入力で与えられる．
for $k = 0$ to N **do**

if $u_0+u_1+u_2=k$ となる，すべての $\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}]$ が 0 **then**
このパイプパズルには解がない．
end if
for $u_0+u_1+u_2=k$ となる，すべての $\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}]$ について **do**
if $\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}]=1$ **then**
for 継ぎ目の向き C : それぞれ $\emptyset,\{T,B\},\{R,L\},\{T,R\},\{R,B\},\{B,L\},\{L,T\}$ について **do**
$\mathrm{add}(\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}],C)$
end for
end if
end for
end for
このパイプパズルには解がある．
}

アルゴリズム add の詳細：アルゴリズム main の中の add は妥当な配置状態にカードを 1 枚追加する関数である．そして，配置状態へカードを追加して，更新した配置状態を返す．このとき add は更新した配置状態の妥当性を検査する．更新した配置状態を検査して，妥当であるなら，対応する配列の要素を 1 とし，妥当でなければ初期値 0 のまま変更しない．

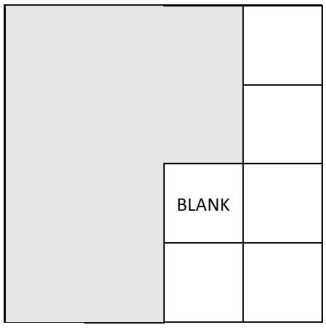
$\mathrm{add}(\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}],C)$\{
if カードを追加するセル (i,j) が最も下の行 $(j=h)$ かつ，C が $\{T,B\},\{R,B\},\{B,L\}$ のいずれかなら **then**
下に継ぎ目を向けるカードの配置は許されないため，
カードを追加できないため関数 add をぬける．
end if
if 配置状態 $\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}]$ へ継ぎ目 C のカードを配置する．このとき図**??**，図**??**にあるパターンなら **then**
if カードの配置のパターンが図**??**と図**??**なら **then**
連結ラベルの特殊な付け替えをする．
$\mathrm{S}[u_0',u_1',u_2',c_1',\ldots,c_{h+1}']=\mathrm{update}(\mathrm{S}[u_0,u_1,u_2,c_1,\ldots,c_{h+1}]])$
else if カードの配置のパターンが図**??**と図**??**なら **then**
新たにひとつなぎである継ぎ目が現れるため，連結ラベルが増える．
else
残ったカードの配置のパターンでは，追加したカードに従って連結ラベルを付け替える．
end if

更新した配置状態 $\mathrm{S}[u'_0, u'_1, u'_2, c'_1, \ldots, c'_{h+1}]$ を作成する．
ただし，$u'_0, u'_1, u'_2, c'_1, \ldots, c'_{h+1}$ はカードを充足領域に追加して，連結ラベルを付け替えたあとのパラメータである．
このとき，前線に付けられている連結ラベルの添え字の数字が連番になるように付け替える．
else
カードを追加できないため関数 add をぬける．
end if
if u'_0, u'_1, u'_2 がそれぞれ d_0, d_1, d_2 を超えていない **then**
$\mathrm{S}[u'_0, u'_1, u'_2, c'_1, \ldots, c'_{h+1}] = 1$
end if
}

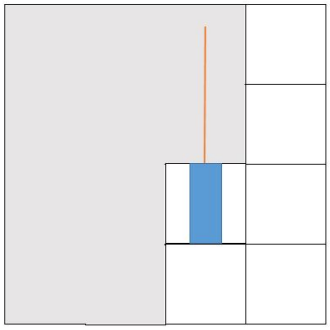
アルゴリズム update の詳細：アルゴリズム add の中の update は，2 本のパイプをつなぐことに伴う，連結ラベルの付け替えをするための関数である．カードを配置するパターンが，図**??**と図**??** のどちらかであるとき，カードの追加によって，2 つの連結ラベルをつなぎあわせて 1 つの連結ラベルへ統一する必要がある．このとき正当性を保つため，連結ラベルに優先度を設ける．それぞれの連結ラベルがもつ優先度は，$l_s < l_1 < \cdots < l_{h'}$ とする．

update($\mathrm{S}[u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1}]$){
カードを追加するセルの座標 (i, j) に対して，c_j, c_{j+1} の値を得る．
$c_j = l_m, c_{j+1} = l_n$ とする．
if $l_m \neq l_s$ かつ $l_n \neq l_s$ なら **then**
妥当性より，
$c_0 = l_m, c_p = l_n$ となる o, p が存在し，$0 < j < j+1 < p$ となっているはずである．
$\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$,
ただし，$c_o = l_{min(m,n)}, c_j = l_B, c_{j+1} = l_B, c_p = l_{min(m,n)}$ とする．
else if $l_m = l_s$ なら **then**
$\mathrm{S}(u_0, u_1, u_2 + 1, c_1, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$
ただし，$c_j = l_B, c_{j+1} = l_B, c_p = l_s$ とする．
else if $l_n = l_s$ なら **then**
$\mathrm{S}(u_0, u_1, u_2 + 1, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$
ただし，$c_o = l_s, c_j = l_B, c_{j+1} = l_B$ とする．
end if
}

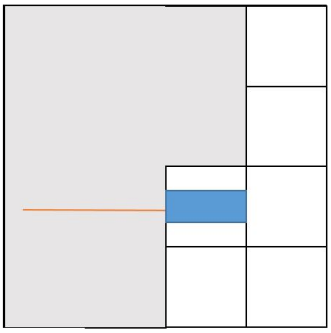
このアルゴリズムでは配置状態を配列で表現した．ここからは配置状態を順序付き集合で表



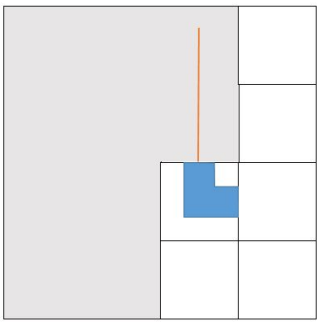
(a) 継ぎ目向き $\emptyset$ で置かれたカード p_0



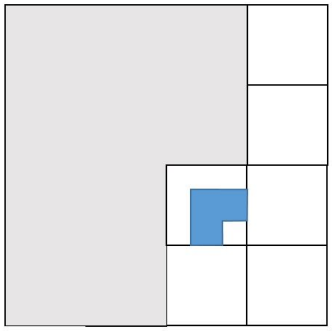
(b) 継ぎ目向き $\{T, B\}$ で置かれたカード p_1



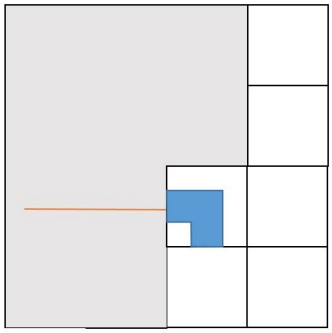
(c) 継ぎ目向き $\{R, L\}$ で置かれたカード p_1



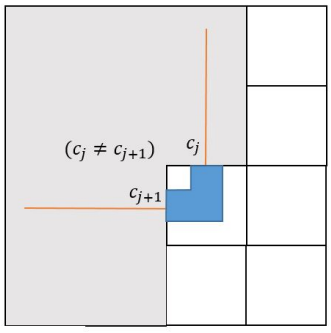
(d) 継ぎ目向き $\{T, R\}$ で置かれたカード p_2



(e) 継ぎ目向き $\{R, B\}$ で置かれたカード p_2

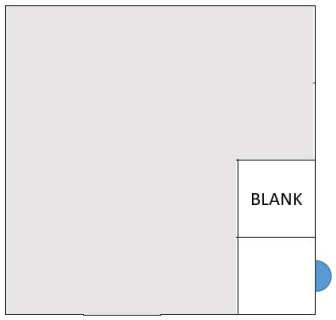


(f) 継ぎ目向き $\{B, L\}$ で置かれたカード p_2

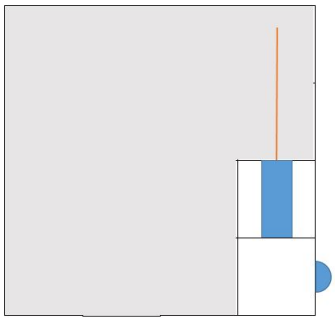


(g) 継ぎ目向き $\{L, T\}$ で置かれたカード p_2

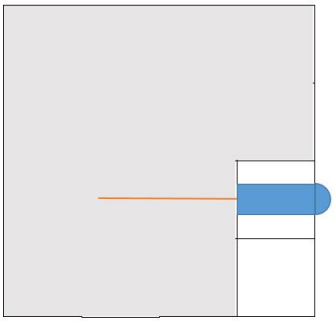
図 4.2: 許されるカード配置．灰色の領域は充足領域とする．そして灰色と白色の間にある線分を前線とする．前線に赤い線が現れている位置が，継ぎ目の位置を表す．この絵では，ひとつなぎのパイプを省略している．カードを配置するとき，上記の継ぎ目の位置のみ許す．



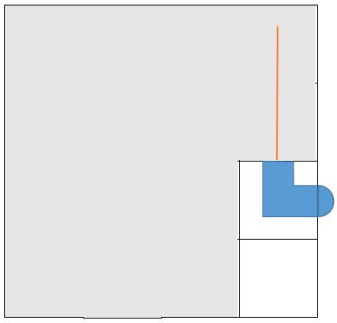
(a) 継ぎ目向き $\emptyset$ で最後の列に置かれたカード p_0



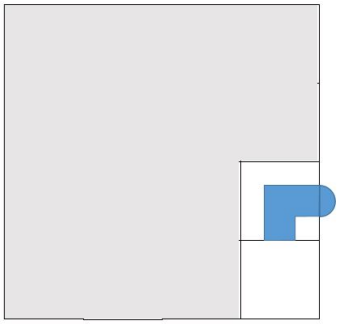
(b) 継ぎ目向き $\{T, B\}$ で最後の列に置かれたカード p_1



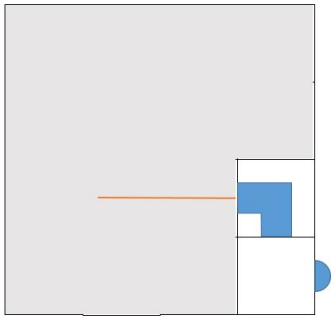
(c) 継ぎ目向き $\{R, L\}$ で最後の列に置かれたカード p_1



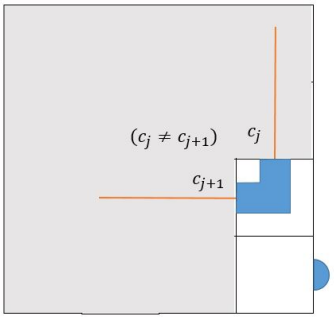
(d) 継ぎ目向き $\{T, R\}$ で最後の列に置かれたカード p_2



(e) 継ぎ目向き $\{R, B\}$ で最後の列に置かれたカード p_2



(f) 継ぎ目向き $\{B, L\}$ で最後の列に置かれたカード p_2



(g) 継ぎ目向き $\{L, T\}$ で最後の列に置かれたカード p_2

図 4.3: 最後の列へカードを追加するとき，許されるカード配置．最後の列ではカードが盤の右辺と隣接する．そのため，配置規則の特別な考慮が必要である．

現し，次の定理を証明する．

定理 2. パイプパズルは高さ h が制限されているときは多項式時間・多項式領域で解ける．そしてその計算時間は $O(h(\frac{h}{2}+2)^{h+1}N^3)$，計算領域は $O((\frac{h}{2}+2)^{h+1}N^3)$ である．

証明：まず，上記のアルゴリズムの正当性を示す．このアルゴリズムは動的計画法に基づいている．その正当性の証明には，配置状態の妥当さを再帰的に示す必要がある．

アルゴリズムにより，いまカードを (i,j) へ追加することを考える．このとき，配置状態が妥当なときのみ，カードの追加処理を実行する．アルゴリズムがカードの追加処理を行うと，更新した配置状態が得られる．u'_0, u'_1, u'_2 がそれぞれ d_0, d_1, d_2 を超えていないと仮定すると，この更新した配置状態が妥当であることを示す．

まず，妥当な配置状態を $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_B$ とする．この配置状態へ追加できるパターンは図??（図??）または図??（図??）である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0+1, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_B$ である．この配置状態は妥当である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0, u_1, u_2+1, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_m, c_{j+1} = l_m$ である．この配置状態は妥当である．

次に，妥当な配置状態を $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_m, c_{j+1} = l_B$ とする．この配置状態へ追加できるパターンは図??（図??）または図??（図??）である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0, u_1+1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_m$ である．この配置状態は妥当である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0, u_1, u_2+1, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_m, c_{j+1} = l_B$ である．この配置状態は妥当である．

次に，妥当な配置状態を $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_m$ とする．この配置状態へ追加できるパターンは図??（図??）または図??（図??）である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0, u_1+1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_m, c_{j+1} = l_B$ である．この配置状態は妥当である．??（図??）の場合，更新後の配置状態は $\mathrm{S}(u_0, u_1, u_2+1, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_m$ である．この配置状態は妥当である．

最後に，妥当な配置状態を $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_j = l_m, c_{j+1} = l_n$ とする．この配置状態へ追加できるパターンは図??（図??）である．このとき，配置状態の種類を 3 つに分ける

- $l_m \neq l_s$ かつ $l_n \neq l_s$ なら，配置状態は $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$，ただし，$c_o = l_m, c_j = l_m, c_{j+1} = l_n, c_p = l_n$ となる．l_m が l_n より優先度が高いとき，$\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$，ただし，$c_o = l_m, c_j = l_B, c_{j+1} = l_B, c_p =$

l_m とし，l_n が l_m より優先度が高いとき，
$\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$，ただし，$c_o = l_n, c_j = l_B, c_{j+1} = l_B, c_p = l_n$ である．これらの配置状態は妥当である．

- $l_m = l_s$ なら，配置状態は $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$，ただし，$c_j = l_s, c_{j+1} = l_n, c_p = l_n$ となる．そのため $\mathrm{S}(u_0, u_1, u_2 + 1, c_1, \ldots, c_j, c_{j+1}, \ldots, c_p, \ldots, c_{h+1})$，ただし，$c_j = l_B, c_{j+1} = l_B, c_p = l_s$ となる．この配置状態は妥当である．

- $l_n = l_s$ なら，配置状態は $\mathrm{S}(u_0, u_1, u_2, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_o = l_m, c_j = l_m, c_{j+1} = l_s$ となる．そのため $\mathrm{S}(u_0, u_1, u_2 + 1, c_1, \ldots, c_o, \ldots, c_j, c_{j+1}, \ldots, c_{h+1})$，ただし，$c_o = l_s, c_j = l_B, c_{j+1} = l_B$ となる．この配置状態は妥当である．

つまり，アルゴリズムは正当であり，パイプパズルに解があることを判定できる．そして端点 s から t への経路を得るには，妥当な配置状態のみを覚えておけばよい．

このアルゴリズムについて，まず多項式領域で計算可能であることを示す．領域量の上界は配置状態がもちうる状態数で上からおさえられる．配置状態は配列 $\mathrm{S}[u_0, u_1, u_2, c_1, \ldots, c_{h+1}]$ で表現した．u_0, u_1, u_2 は 0 から N までの値を取る．$c_1, \ldots, c_{h+1}$ は，$l_1, \ldots, l_{h'}$ と l_s, l_B を取りうる．ただし，h' は解をもちうる前線だけを考えているので，$h' \leq h/2$ が成立する．これらにより，状態数は $O((\frac{h}{2} + 2)^{h+1} N^3)$ となり，多項式領域である．

さらにこのアルゴリズムは多項式時間で計算可能であることを示す．アルゴリズムの計算時間は，カード追加処理にかかる時間とその処理回数によって得られる．カード処理は，図??，図??であるとき，他よりも処理に時間がかかる．しかし，高さ h に対する処理のため，$O(h)$ 時間で行える．その他のカード追加処理の計算時間は定数時間である．カードの追加処理に伴い，連結ラベルの添え字を逐一連番に付け替える処理が発生する．これは高さ h に対する処理である．この時間は $O(h)$ である．そのため，アルゴリズムの計算時間は $O(h(\frac{h}{2} + 2)^{h+1} N^3)$ となる．つまり，このアルゴリズムは多項式時間で計算可能である． □

系 1. パイプパズル問題を解く，高さ h に関する Fixed Paramter Tractable[1]アルゴリズムが存在する．

4.2 高さに制限がないパイプパズル

高さに制限のないパイプパズルでは，盤の高さ h を制限せず，ブランクカード p_0 を何枚でも使えるものとする．そしてパイプカード p_1, p_2 は合計 N 枚とする．

[1]ある問題 P に対して，長さ n の入力とパラメータ h が与えられたとき，この問題 P に対する実行時間 $O(f(h)p(n))$ 時間のアルゴリズムが存在するならば，このアルゴリズムを h に関して Fixed Parameter Tractable なアルゴリズムであるという．ただしここで，$f(h)$ は h に関する任意の関数で，$p(n)$ は n に関する任意の多項式関数とする．

扱う盤の高さに制限がないため，ある行を 0 行目として，盤の行数を下へ向かって $1, 2, \ldots$ 行目として，上へ向かって $-1, -2, \ldots$ 行目とする．

このパイプパズルでは，特定の入力のとき，線形時間で解けることを示す．

定理 3. 幅 $w = 1$ のとき，高さに制限のないパイプパズルは線形時間で解ける．

証明：このとき，端点の相対位置によって分けてパイプパズルを議論する．

まず，端点 s, t が同じ行に配置されている場合について議論する．端点 s, t が同じ行に配置されているなら，その解は直線パイプカードの枚数 d_1 が $d_1 = 1$ であるときのみである．それ以外のとき解はないため，端点が同じ行であるときパイプパズルは定数時間で解ける．

次に，端点 s, t が異なる行に配置されている場合について議論する．端点 s, t が同じ行に配置されていないなら，その解には必ず，直角パイプカードの枚数 d_2 が $d_2 = 2$ である必要がある．これに加え，端点の相対的な位置によって p_1 が必要になる．j, j' がそれぞれ s, t の設置されている行だとする．このとき，直線パイプカードの枚数 d_1 は $d_1 = |j - j'| - 1$ が必要になる．

よって，端点が異なる行に配置されているとき，$d_1 = |j - j'| - 1, d_2 = 2$ なら解があり，それ以外なら解がない．つまり，パイプパズルは線形時間で解ける．

よって幅 $w = 1$ のときパイプパズルは線形時間で解ける．

□

定理 4. 幅 w が $w = 6 + 2k\,(k = 0, 1, 2, \ldots)$ かつ，それぞれの端点 s, t の設置される行 j, j' が $|j - j'| = 0$ であるとき，高さに制限のないパイプパズルは線形時間で解ける．

証明：盤の幅が 6 以上の偶数で，かつ端点 s, t が同じ行に配置されているとき，このパイプパズルは線形時間で解けることを示す．

入力のカード枚数について線形時間で解けることを示す．そのため，解がない場合を条件別に次に示す．すべての条件を満たさないときのみ，解があることを示す．

1. $d_1 + \frac{d_2}{2} < w$ **のとき解はない：**

   なぜなら，幅 w の端点同士をつなぐためのパイプがなければ，パイプパズルに解はないからである．端点 s, t をひとつなぎにするためには，現在いる行を維持しながら，s から右へ w 個，列を進む必要がある．

   p_1 は，1 枚使うことで列を最大 1 つ進む．これは明らかである．それに対して，p_2 は，4 枚使うことで列を 2 つ進む．なぜなら，列を進み，元の行に戻るためには，p_2 を最低 4 枚つなげる必要があるからである．p_2 を 4 枚つなげると列を 2 つ進むことができる．これにより，$d_1 + \frac{d_2}{2} < w$ のとき解はないことがわかる．$d_1 + \frac{d_2}{2} = w$ のとき，必ず最短で経路を構成しなければならない．

   u_1, u_2 をそれぞれ現在まで使っている p_1, p_2 の枚数とする．$u_1 + \frac{u_2}{2} = w$ となる u_1, u_2 で構成される配置を，**タイトな配置**と呼ぶ．

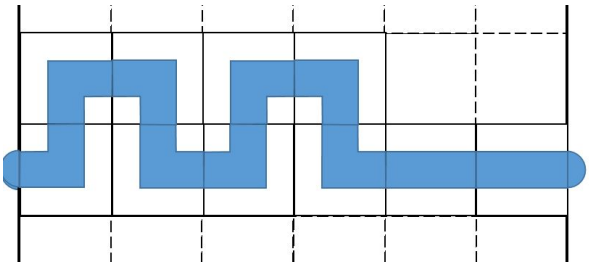


図 4.4: $u_1 = 2, u_2 = 8$ のタイトな配置の例.

2. **p_1, p_2 がどちらも偶数でなければ解はない：**

   まず，p_2 が奇数であるときを考える，このとき p_2 は 1 枚で 90° の方向転換をしなければならない．このため，すべてのパイプカードを敷き詰めても継ぎ目が上または下を向いている状態になってしまう．よって，p_2 が奇数であるとき解はない．

   次に，p_1 が奇数，p_2 が偶数であるときを考える．このとき，$d_1 + \frac{d_2}{2} = w$ と $d_1 + \frac{d_2}{2} \geq w$ で分けて議論する．

   - $d_1 + \frac{d_2}{2} = w$ であるとき
     このとき，パイプパズルはタイトな配置になるので，p_1 は必ず $\{R, L\}$ で端点の間に配置しなければならない．そして，s, t 間の残りのセルを p_2 のみでつながなければならない．このとき p_1 が奇数枚であるため，残ったセルは奇数である．しかし，これを p_2 のみでつなぐことはできない．なぜなら，p_2 のみを使って，同じ行の右と左へ継ぎ目を向けるとき，セルは必ず横方向へ 2 つずつ埋めてしまうからである．以上のことから，解はないことがわかる．
   - $d_1 + \frac{d_2}{2} \geq w$ であるとき p_1 と p_2 でタイトな配置を構成する．このとき，そこに使われない p_1 がでる．そして，その枚数は必ず奇数になる．なぜなら，前述のとおり，タイトな配置に使われる p_1 は，必ず偶数でなければ，経路が構成できないからである．タイトな配置に使われない p_1 は，タイトな配置に使われている p_2 と組み合わせて，図**??**に示すパイプの迂回路を構成しなければ配置できない．しかし，この迂回路を構成するには，p_1 が偶数枚必要になる．なぜなら，p_1 が奇数枚であるとき，迂回路の往路は構成できても，復路を構成できないからである．

   よって，p_1, p_2 がどちらも偶数でなければ解はない

3. **$d_2 = 2$ なら解はない：**

   なぜなら p_2 が 2 枚のみでは，パイプを端点のある行へもどせないからである．そのため，このとき解はない．

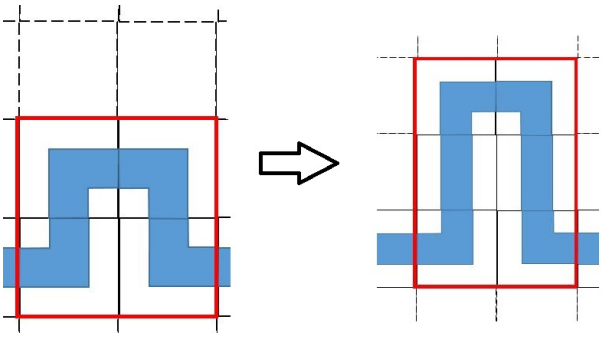


図 4.5: p_2 へ p_1 を組み合わせて構成する迂回路．p_1 を偶数枚ずつ追加するなら配置できる．

4. **入力のカードがすべて p_1 であるとき，$d_1 \neq w$ なら解はない：**

   なぜなら，p_2 がなければ，盤の縦方向にパイプを旋回することができない．そのため，$d_1 = w$ でなければ解はない．

5. **入力のカードがすべて p_2 であるとき，$d_2 \neq 2w + 8k\,(k = 1, 2, \dots)$ なら解はない：**

   タイトな配置を構成するとき，$d_1 = 0$ なら p_2 が $2w$ 枚必要となる．タイトな配置に使われない枚数 $d_2 - 2w$ は，8 枚ずつ現在の配置に追加して迂回路を構成する．そして，それ以外に追加方法はない．なぜなら，2 枚の p_2 で構成された凸部分へ p_2 を追加する方法しかないからである．その迂回路を図**??**に示す．2 枚の p_2 で構成された凸部分をカード回転により，それぞれ $\{R, B\}$ から $\{B, L\}$，$\{B, L\}$ から $\{R, B\}$ とし，p_2 を追加できるようにする．すると．元の経路にもどるためには，p_2 が 8 枚必要であることがわかる．

   そして，この迂回路は $w \leq 4$ なら構成できない．なぜなら，この迂回路を構成するには，p_2 の凸を中心に，幅が左右に 1 つずつセルが必要だからである．$w = 1, 2, 3$ では，迂回路を構成するには幅が足りない．$w = 4$ では，p_2 の凸位置が 1 列目 2 列目に，または 3 列目 4 列目にまたがる．このとき，右または左へパイプカードを配置できないため迂回路を構成できない．

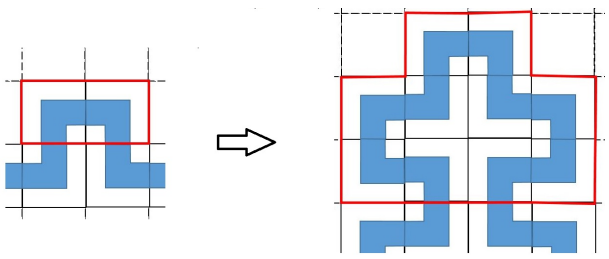


図 4.6: p_2 に対して新たに p_2 を 8 枚追加し，迂回路を構成する方法．

上記の条件をすべて満たさないとき解がある：
ここまでの条件をすべて満たさない入力を，3 つに分ける．

1. $d_1 = w$ かつ $d_2 = 0$.
2. $d_1 = 0$ かつ $d_2 = 2w + 8k\,(k = 0, 1, 2, \dots)$.
3. $d_1 = 2 + 2k\,(k = 0, 1, 2, \dots)$，$d_2 = 4 + 2l\,(l = 0, 1, 2, \dots)$，かつ $d_1 + \frac{d_2}{2} \geq w$.

このそれぞれについて，解があることを示す．

1. $d_1 = w$ **かつ** $d_2 = 0$：

   このときすべての p_1 を $\{R, L\}$ で配置すれば，s, t の経路が構成できる．つまりこの入力に対してパイプパズルは解がある．

2. $d_1 = 0$ **かつ** $d_2 = 2w + 8k\,(k = 1, 2, \dots)$：

   このとき，p_2 を $2w$ 枚使って，タイトな配置を構成できる．タイトな配置に使われない $8k$ 枚の p_2 は，配置されている p_2 と組み合わせて，図**??**に示す迂回路を構成できる．この迂回路は p_2 が 8 枚ずつで構成できる．つまりこの入力に対してパイプパズルは解がある．

3. $d_1 = 2 + 2k\,(k = 0, 1, 2, \dots)$，$d_2 = 4 + 2l\,(l = 0, 1, 2, \dots)$，**かつ** $d_1 + \frac{d_2}{2} \geq w$：

   このとき，$d_1 + \frac{d_2}{2} \geq w$ を満たすため，s, t 間の経路を構成できる．この経路の配置がタイトな配置なら，使われていない p_1, p_2 が偶数枚ある．この p_1, p_2 が偶数枚あっても，迂回路を構成できることを議論する．

   まず，偶数枚の p_1 で迂回路を構成できることを示す．これは，p_2 が与えられていることと，図**??**より明らかである．

   次に，p_2 が偶数枚でも迂回路を構成できることを示す．p_2 のみが与えられていたとき，p_2 を使った迂回路の構成には p_2 が 8 枚必要だった．ここでは，p_1 が一緒に与えられれば，p_2 が偶数枚でも迂回路を構成できることを示す．

   - p_2 **を 4 枚追加して迂回路を作る方法：**
     タイトな配置に p_2 を 4 枚追加して迂回路を構成する方法を図**??**に示す．まず，2 枚の p_1 が連続する箇所を取り除く．その取り除いた箇所へ 2 枚 p_2 を継ぎ目の向き $\{L, T\}, \{T, R\}$ で配置する．上を向いた 2 つの継ぎ目に対して，取り除いた p_1 を $\{T, B\}$ で配置する．さらに出てきた継ぎ目に対して，2 枚の p_2 を継ぎ目向き $\{R, B\}, \{B, L\}$ で配置する．すると，p_2 を 4 枚追加して迂回路が作れる．

- **p_2 を 6 枚追加して迂回路を作る方法：**

  タイトな配置に p_2 を 6 枚追加して迂回路を構成する方法を図**??**示す．この迂回路は，図**??**から構成される迂回路を基準とし，図**??**に示す，p_2 を 2 枚追加する方法を行う．すると，6 枚の p_2 から迂回路を構成する．

- **（別方法の）p_2 を 8 枚追加して迂回路を作る方法：**

  上述の方法とは異なる方法で，タイトな配置に p_2 を 8 枚追加して迂回路を作る方法を図**??**に示す．この迂回路は，図**??**から構成される迂回路を基準とし，図**??**に示す，p_2 を 2 枚追加する方法を行う．すると，8 枚の p_2 から迂回路を構成する．

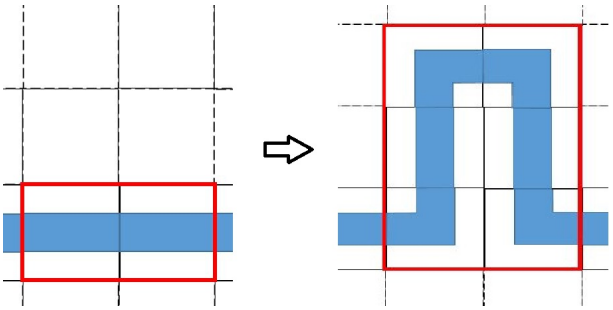
- **p_2 を 10 枚以上追加して迂回路を作る方法：**

  ここからは，いままで示した迂回路を組み合わせることで，タイトな配置に p_2 が 10 枚以上追加されても迂回路を構成できることを示す．上記の方法より，2 枚の p_1 が連続する箇所を起点として，$4, 6, 8, 10$ 枚の p_2 で迂回路が構成できる．

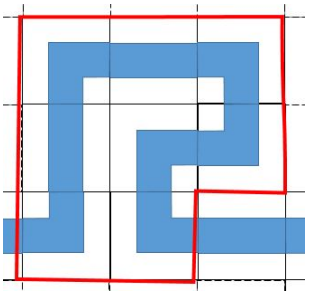
  さらに，それぞれの迂回路構成を組み合わせることで，$12, 14, 16, ...$ 枚の p_2 で迂回路を構成できる．例えば p_2 が 12 枚の迂回路を構成するなら，図**??**の色付けした 2 枚の p_1 に対して，新たに p_2 を 4 枚追加して図**??**を構成できる．さらに例を挙げる．p_2 が 14 枚の迂回路を構成するなら，図**??**へ図**??**を組み合わせることで構成できる．つまり，8 枚構成の迂回路を基準として，p_2 が偶数枚追加されても迂回路が構成できることがわかる．

以上の 3 つについて解があることを示すことができた．そして，それ以外では解がないことを示した．つまり，幅 w が $w = 6 + 2k\,(k = 0, 1, 2, \dots)$ かつ，端点 s, t 設置される行 j, j' が $|j - j'| = 0$ であるとき，高さに制限のないパイプパズルは，線形時間で解けることが証明できた．

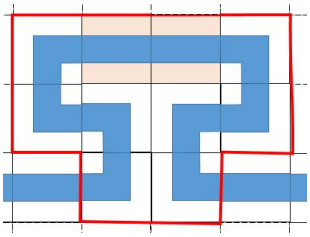
□



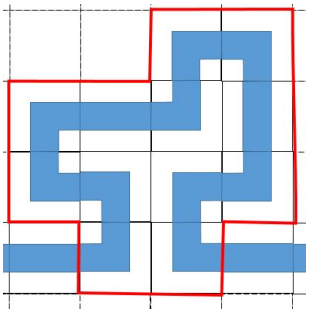
(a) p_2 を 4 枚追加して構成する迂回路.



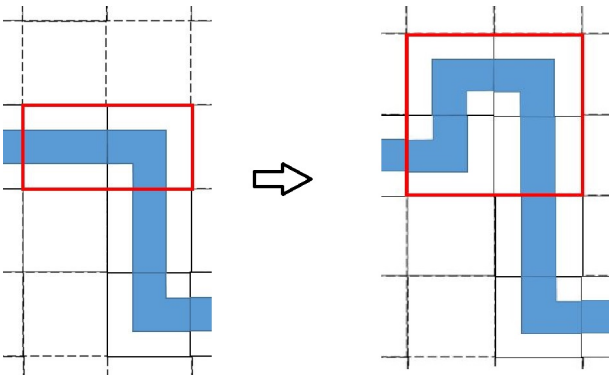
(b) p_2 を 6 枚追加して構成する迂回路.



(c) p_2 を 8 枚追加して構成する迂回路.



(d) p_2 を 10 枚追加して構成する迂回路.



(e) 既に配置している p_1, p_2 を基準に p_2 を 2 枚追加して構成する迂回路.

図 4.7: 迂回路の構成

参考文献

[1] Robert A. Hearn and Erik D. Demaine, *Games, Puzzles, and Computation*, Cambridge, 2009 (邦訳「ゲームとパズルの計算量」, 上原隆平訳, 近代科学社, 2010).

[2] Erik D. Demaine and Martin L. Demaine, Jigsaw Puzzles, Edge Matching, and Polyomino Packing: Connections and Complexity, *Graphs and Combinatorics*, Vol. 23, Supplement 1, pp.195-208, 2007.

[3] Jeffrey Bosboom, Erik D. Demaine, Martin L. Demaine, Adam Hesterberg, Pasin Manurangsi, Anak Yodpinyanee, Even $1 \times n$ Edge-Matching and Jigsaw Puzzles are Really Hard, *Journal of Information Processing*, Vol.25, pp.682-694, 2017.

[4] Michael R. Garey and David S. Johnson, *COMPUTERS AND INTRACTABILITY : A Guide to the Theory of NP-Completeness*, W. H. FREEMAN AND COMPANY New York, p.96, 1979.