データベースシステム 98－4
（１９９４．５．２７）

# 空間検索のためのハッシュデータ管理

宮崎雅子　　三浦孝夫
埼玉大学　　産能大学

本報告では領域情報を扱い，与えられた点を含む矩形をO(1)で検索するという質問に対するアルゴリズムを提案する。領域情報は座標上で広がりをもつ範囲として表現され，矩形も領域情報に含まれる。本報告のデータ表現法はデータを削除挿入しても検索の計算量はO(1)に保たれる。

# Hash Data Management for Spacial Search

Masako MIYAZAKI and Takao MIURA
Saitama University　SANNO Colleag

In this investigation, we propose a sophisticated algorithm by which any rectangle containing a given point can be obtained in O(1). Here we assume regions are described by a set of rectangles, in particular, a rectangle itself is a kind of region. Data representation proposed here is robust in a sense that we can still keep O(1) property against insertion and deleteion.

## 1 まえがき

領域情報は幾何情報と文字数値情報で構成される。例えば，世界地図では，幾何情報は国の位置，形状を表し，文字数値情報は国の名前，人口，主な産業などを表現する。本報告では主に，幾何情報を扱い，幾何情報から文字数値情報を得るためにハッシュ法を使う。

本報告では，与えられた互いに交わらない矩形集合に対して，質問として次の２つを考えている。

1. 与えられた点を含む矩形を検索する。

2. 与えられた図形を含む矩形を検索する。

例として，図１を考える。図１の座標に点Ｐを与える。点Ｐを含むような矩形を検索する。例では，矩形Ａが求める解となる。

例えば，広域地図では，病院や学校などは点で表される。神奈川県の地図があるとする。都市は矩形として考えられる。点で表されている病院を含む都市を検索するのに，本報告のデータ表現を使うと一定時間でできる。

矩形を表現する方法は大きくわけて２つ挙げられる。

1. 矩形の境界線を管理する方法。

2. セル法。
   領域を分解し，メッシュ状に分割して管理する方法。

本報告では矩形を分割していく方式を採用している。ここでは、矩形を含む領域全体を２等分割し、分割してできた長方形が矩形に含まれるまで，Ｘ軸方向，Ｙ軸方向にそれぞれ交互に２等分割していく。ここで，Ｘ軸Ｙ軸がそれぞれ，Ｎビットの２進数で構成されていると考える。２等分するということは，上位のビットが０から１へ変化することであり，２等分割を２進数で表現することができる。

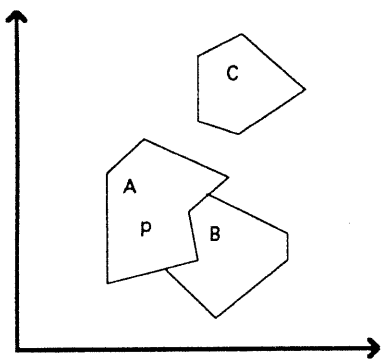

図1　点pを含む矩形

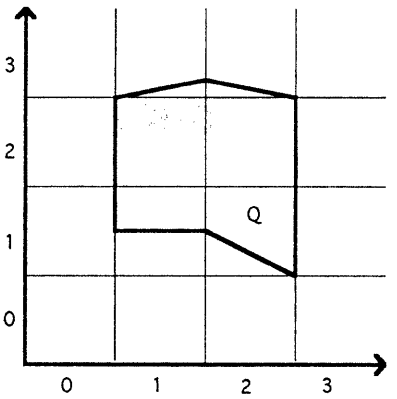

図2　ブロック法

検索アルゴリズムでは与えられた点を含む長方形を検索する。長方形を検索するパスは２通りしかなく，検索する長方形の個数も定数個となるため，検索アルゴリズムの計算量は一定となる。

本報告の構成を述べる。２章では矩形情報における他のデータ表現方法について述べる。３章で本報告で提案するデータ表現方式について述べ，それを評価し，まとめる。

## 2 矩形情報の表現法の概要

### 2.1 １次元配列の表現法

空間内の矩形情報を管理する方法として，まず考えられる方法は，矩形を矩形内の総ての点で表現し，１次元配列とする管理方式である。しかし，この管理方式では，座標上の全ての点を情報として持たなくてはならない。ハッシュを使えば計算量は一定になるが，記憶領域が膨大となり，現実的でない。

そこで、矩形情報を等間隔のブロックで分割する方法を考える。

矩形情報ごとに等間隔のブロックで分割し、ブロックの情報をハッシュして，ハッシュテーブルに入れる。図２の例では，矩形を分割し，それぞれ０から３までのブロック番号をつけている。点Ｑはブロック番号(2,1)に属している。

与えられた点を含む矩形は、点を含むブロックを計算し、求めたブロックをハッシュして求められる。

しかし，この方式では，ブロックの細かさに情報の正

確さが依存する。また、ブロックすべてに矩形の情報を持つため，ブロックを細かくすると，それだけ記憶域を必要とする。

## 2.2 R木

R木では、点と長方形を効率よく管理することを目的としている。各ノードはM個のindex recordを持ち、その半分が埋まっていなければならない。index recordにはleafでない限り長方形の情報を持っている。leafノードには実際のデータが入っている。ノード中の全ての長方形、またはデータを含む最小の長方形を親ノードが持つ。

図3は空間のデータをR木で管理している。R15に実際のデータが入っている。R6はR15とR16を含んでいる。R2はR7とR6を含む。この図ではR2はrootとなり、R6とR7を子供として持っている。R6はさらにR15とR16を子供として持つ。

それを木構造になおすと図4となる。

R木は計算量が木の深さに依存する。メモリーも，葉ノード以外にはデータが入ってないので，木の内部点として無駄に使っている。

本報告の質問をR木で考える。R木は区画がデータである。点を含む矩形を検索する場合，木のrootからデータを持つleafノードまで，木を辿る必要がある。データ数をNとおくと，$\log N$の計算量となってしまう。

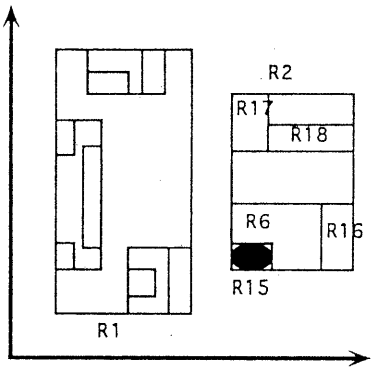

図3 R木

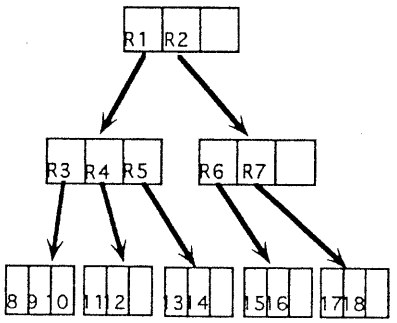

図4 R木の木構造

## 2.3 4分木

4分木は空間上の像を分割し表現する。2次元空間を例に考える。2次元空間は説明しやすくするため$2^n*2^n$の大きさとする。

まず，縦横で2等分する。$2^{n-1}$の4つの長方形ができる。この長方形が矩形に含まれれば領域分割は終了する。含まれない場合，長方形を縦横で分割する。この分割過程を木構造で表現する。ノードには分割する長方形の情報と，子供へのポインタが入っている。leafノードには矩形を表す情報が入っている。

図5の例を考える。全体を4等分に分割し、分割して生じた長方形が矩形Aに含まれるまで，分割は続く。

それを木構造に直したものが図6である。ノードpには長方形Pの情報と，4つの子供へのポインタが入っている。leafノードには，矩形Aかそうでないかの情報がはいっている。

矩形の検索は木をrootからたどることで行われる。そのため，検索の計算量は木の深さに依存する。

分割されできる長方形の数とleafノードの数は等価である。このため，木の深さは長方形の数に依存する。長方形の数は，矩形の形の複雑さに比例する。また，メモリーは長方形の数だけ必要となる。

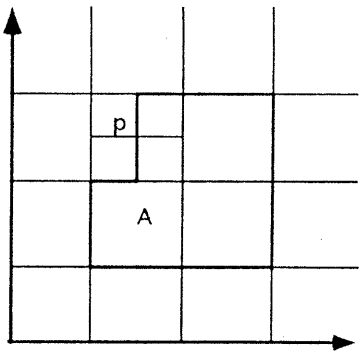

図5 4分木

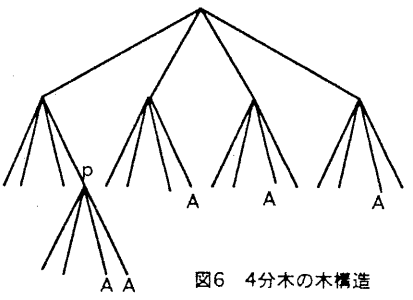

図6 4分木の木構造

## 2.4 境界線のベクトル表現

矩形情報として，境界線をベクトル表現する。ベクトルの向きを固定する。与えられた点が矩形に含まれるかという質問に対して，点がベクトルの左右どちらににあるか計算することで，内点かどうか判断できる。

図7を考える。ベクトルの向きは時計回りで与えられている。点pが与えられている。点pはベクトルの右側にあり，矩形の内点であることが判断できる。

この方法では，与えられた点の内外判定は，各矩形ごとに計算する必要がある。このため，計算量は，矩形の数に依存してしまう。

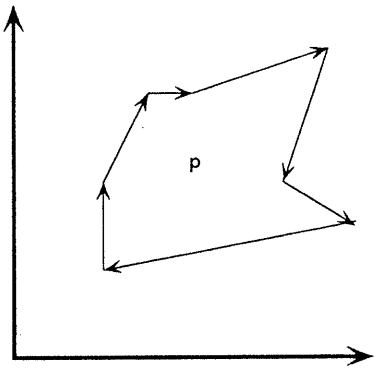


図7 矩形のベクトル表現

# 3 情報の操作

与えられた点と図形を含む矩形を求めるアルゴリズムを提案する。本報告のアルゴリズムはデータ数に関係なく，計算量O(1)で検索を行う。またメモリーは矩形を分割する長方形の数だけ持てばいい。

## 3.1 矩形情報の表現

区画情報を表現する方法を述べる。区画情報は境界線で区切られた互いに素な矩形領域である。矩形領域は直線分の境界線で表現されている。線分は2つの端点で表される。

まず矩形の領域をX軸Y軸で交互に2等分割していく。2等分割の利点を考える。

1. 2等分割は2進数で表現できる。
2. 2進数のビット数は定数である。
3. ビット数が定数であるため，分割回数も定数となる。
4. 矩形を分割する長方形も定数となる。
5. 計算量が一定となる。

この結果，矩形は大小の，全く重ならない長方形で表現できる。例として図8を考える。X軸は0から$2^{M_1}$まで，Y軸は0から$2^{M_2}$までの，領域Sを考える。

1. X軸方向に領域を2等分割する。分割点は$2^{M_1-1}$となる。
2. 分割された領域が矩形に含まれない場合，Y軸方向に分割する。
3. 長方形の領域が矩形に含まれるまで分割を繰り返す。

分割点が$2^{M_1-1}$となることを2進数で表現することは，上位1桁目が1から0となることと等価である。

このようにして，図形AがN個の長方形で表現できる。

また長方形は次の項目で表現できる。

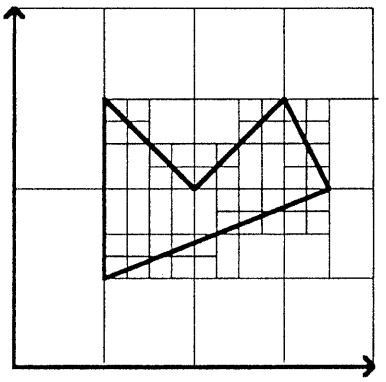

図8 矩形の分割の様子

- 最小点のX座標
- 最小点のY座標
- 縦の長さ
- 横の長さ

この時，無限回の分割を防ぐため，縦と横の長さは1以上となるようにする。

N個の長方形の情報をハッシュテーブルにまとめる。本報告ではハッシュ関数は与えられているものと考える。

## 3.2 矩形情報の検索

### 3.2.1 点位置決定問題

図形を求める問題は，その点を含む矩形を検索することと等価になる。

点pが与えられたとする。点pは長方形の内部の点である。長方形を表現している点ではないため，自分が含まれている長方形を求める必要がある。点pを含み，点pに一番近い長方形は，長方形の最小のサイズを持つ。

例で考える。図9では長方形で分割されている矩形に点pが与えられている。簡単のため，1つの矩形だけで考える。この場合，点pを含む長方形Rを求める。長方形Rをハッシュすることによって，矩形情報が得られる。

そのため，以下を行う。

1. 点pを含む最小の領域（長方形）を求める。
2. 求めた点をハッシュする。
3. ハッシュテーブルに含まれれば，終了する。
4. 含まれなければ，X軸(またはY軸)方向に求めた領域を含む最小の領域を求め，ハッシュする。
5. ハッシュテーブルに含まれない時，Y軸(またはX軸)方向に次に小さい領域を求める。
6. 含まれるか，(x,y)が0になるまで繰り返す。

領域分割の時，最後の分割がX軸で行われたのか，Y軸で行われたのか，検索では不明である。このため，まずX軸で分割が終了したと考える。X軸方向で領域を求める。X軸の分割の前は必ずY軸で分割しているので，Y軸方向で領域を求める。このようにX軸とY軸と交互に領域を求めていく道と，同様にY軸とX軸と交互にと領域を求めていく道と2通りしか探索パスはない。

点pを含む最小の領域を求める方法を述べる。

矩形情報を作成した時，分割した回数の最大数をkとおく。分割回数が多いほど分割される長方形は小さくなるため，kは最小の長方形のサイズを表している。

1. 点pのX座標，Y座標を2進数で表現する。
2. 点pの上位k桁から下を0にする。
3. 以上で求められた点が，最小の長方形の情報である。

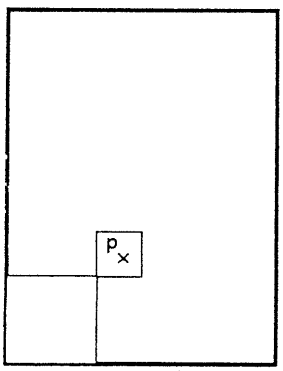


図9　長方形の検索

4. 次に小さい長方形は，k桁目を0にすることで求められる。

コンピュータでは，1語(word)を単位と考える。1語は16ビットか，32ビットで表現される。ビット長が有限であることから，x方向の計算量は一定となる。y方向も同様であるので，全体として一定となり，全体の計算量はO(1)となる。

挿入削除がおこる動的なデータに対しても，検索アルゴリズムは一定時間でできる。

### 3.2.2　領域位置決定問題

与えられた図形を含む，または重なる矩形を検索するアルゴリズムを提案する。図形は長方形に限る。

図10の例で示したような矩形を全てもとめる。(a)の例は図形は矩形に含まれている。与えられた図形Pは矩形Aに含まれているので，矩形Aが求める解となる。(b)の例では図形は複数の矩形と重なっている。与えられた図形Qは矩形A，矩形Bと重なっている。解として矩形Aと矩形Bが求められる。

与えられた図形を分割アルゴリズムでできた長方形で分割する。求めた長方形をハッシュして，含まれる，または重なる長方形を求める。

まず，図形の境界線の長方形を求める。

1. 図形の最大値を含む長方形を求める。
2. 長方形と図形との交点を求める。
3. 交点を含む長方形を求める。
4. 以上を繰り返す。

例えば、図11では、図形Pが与えられている。図形Pの頂点pを含む長方形Rを求める。図形Pと長方形Rとの交点a,bを求める。交点を含む長方形をそれぞれ求める。図形Pの4辺と交わっている長方形をすべて求め，求めた長方形をハッシュし，矩形を求める。

境界線上の長方形の情報だけでは，図形に含まれる矩形を求めることができない。求めた長方形をハッシュし，矩形を求める。計算量は求める長方形の数に依存する。

## 3.3　矩形情報の更新

与えられた矩形情報は互いに交わらない。このため，矩形情報の挿入，削除は簡単に行うことができる。

矩形情報は，挿入前の段階で矩形情報の与えられていない場所へ挿入される。また，矩形を分割するアルゴリズムは矩形情報ごとに行う。このため，挿入は，挿入された矩形情報を情報表現のアルゴリズムで分割することにより実現される。挿入アルゴリズムによって，検索アルゴリズムで調べる長方形の数は変化しない。このため，矩形検索の計算量は一定である。

削除はハッシュテーブルから削除する矩形のデータを削除することによって実現される。削除アルゴリズムも挿入と同様に，検索アルゴリズムの計算量に影響しない。

### 3.4 長方形の合併

本方式では，データ量は長方形の数となる。このためデータ量を減らすため，長方形を合併する必要がある。

長方形は，分割のアルゴリズムから明らかに，自分と同じ大きさの長方形を，左右上下の4方向のうち，1方向に持つ。そういう意味で，対となっている長方形A，Bが存在するとする。今，長方形AとBの情報が，ハッシュテーブルにある。この時，長方形Aの情報を削除し，長方形Bの大きさに長方形Aを足すことにより，データ量を減らすことができる。

検索において，削除された長方形Aの情報にたどりつけば，その探索パス上に長方形Bの情報がある。

しかし，この方式では，データ量が減る反面，探索パスが長くなってしまう。

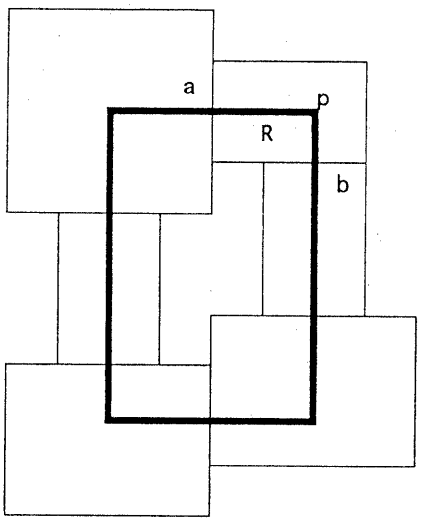


図11 領域位置決定

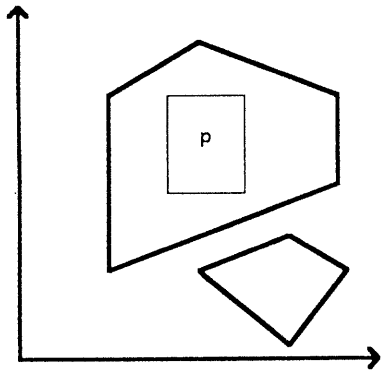


図10(a) 矩形に含まれた長方形

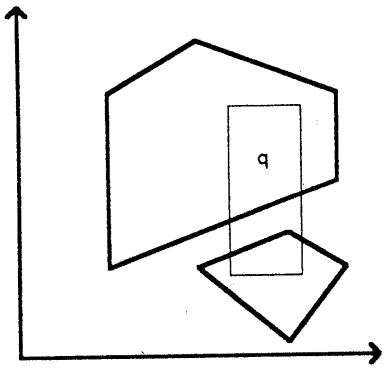


図10(b) 複数の矩形と重なった長方形

## 4 実験による評価

### 4.1 実験

実験として，次の2種類のデータに対して，10万件のデータで検索をおこなう。

1. 矩形情報を分割したデータ
2. 長方形を合併したデータ

本稿では，ハッシュ関数hとして，以下の関数を使う。

$$h = (x + y) \bmod q$$

x,yは長方形の頂角の座標を示しており，qには素数が入る。

しかしここで重要なことは，ハッシュ関数ではなく，ハッシュテーブルにアクセスした回数である。以下に，実験結果として，ハッシュテーブルにアクセスした回数を示す。単位はアクセス回数が回、データ量が件である。

データは各国の地図を用いる。

### 4.2 考察

1件の検索に対するアクセス回数の平均は、日本のデータが42回、フィリピンが16回、韓国が15回、北朝鮮が18回、香港が5回、シンガポールが7回となった。

香港のアクセス回数はシンガポールの0.73倍となった。この2つの地域のデータ量を比較すると、アクセス回数の少ない香港のほうが1.9倍データ量が多い。一方、

| 国名 | | アクセス回数 | データ量 |
|---|---|---|---|
| 日本 | 合併前 | 4175840 | 4749 |
| | 合併後 | 4205937 | 3070 |
| フィリピン | 合併前 | 1626118 | 26341 |
| | 合併後 | 1654429 | 12369 |
| 韓国 | 合併前 | 1462500 | 5557 |
| | 合併後 | 1481250 | 2624 |
| 北朝鮮 | 合併前 | 1791080 | 8611 |
| | 合併後 | 1809644 | 4187 |
| 香港 | 合併前 | 531253 | 639 |
| | 合併後 | 703125 | 271 |
| シンガポール | 合併前 | 734378 | 339 |
| | 合併後 | 918750 | 141 |

韓国と北朝鮮を比較すると、北朝鮮の方がアクセス回数が1.2倍、データ量が1.6倍多い。

この結果、データ量の多い地域のアクセス回数が多いということは言えない。アクセス回数は分割してできる長方形の大きさに依存することが考えられる。

また、日本のアクセス回数が他のケースに比べて多い。特に合併前のデータにおいて、フィリピンと比べて、データ量は日本の方が5.5倍増加している。これに対し、アクセス回数は0.39倍フィリピンのほうが少ない。

これは、日本のデータがほとんど海領域となり、1つの長方形が大きくなってしまったという原因が考えられる。これに対してフィリピンを含む他のデータは矩形が一様に分布している。従って、本報告でのアルゴリズムは細かいデータの表現に優れていると言える。しかし、データの形にその性能が依存する。

合併のアルゴリズムにより，データ量は大幅に減った。アクセス回数は，予想通り，合併前の方が合併後より少ない。北朝鮮の実験結果を見ると、データ量は0.49倍と合併により減少しているが、アクセス回数は1.01倍と増加している。しかし，データ量の減り方に比べ，アクセス回数はそれほど減っていないと言える。

以上から，減ったデータ量に比べアクセス回数の増加は少ないという結果を得た。

## 5 結論

本報告で，点を含む矩形を検索する問題が，アルゴリズムの上では一定時間でできることを示した。このアルゴリズムは，静的データだけでなく，動的データにも，対応できる。

分割してできる長方形の数は，矩形情報の形の複雑さに比例する。しかし，点位置決定問題では，4分木と違い，長方形の数が矩形検索の計算量に影響しない。4分木では計算量は木の深さに依存し，木の深さは長方形の数で決定する。本報告の検索アルゴリズムでは，検索パス上の長方形の数が1つであるため、計算量は一定となる。このことを実験により示した。

しかし、本報告のアルゴリズムは検索の性能がデータの形に依存する欠点を持つ。今後、データの形に検索の性能が左右されないアルゴリズムへの発展が期待される。それとともに、データである長方形を減少させることも期待される。

## 参考文献


[1] 坂内正夫，大沢裕共著，画像データベース，昭晃堂

[2] A.Guttman,R-TREE:A DYNAMIC INDEX STRUCTURE FOR SPATIAL SEARCHING(1984)

[3] Nobert beckmann,Hans-Peter Kriegel The R*-tree:An Efficient and Robust Access Method for Points and Rectangles+