Examination 解説

細川　寛晃

問題概要

- N人の学生が数学と情報の試験を受ける
  - 数学はS_i点、情報はT_i点

- 数学、情報、合計点のすべてで合格基準を超えていれば合格

- Q回のクエリに答える:
- 「数学がX_j点以上かつ情報がY_j点以上かつ合計点がZ_j点以上の学生(合格者)は何人いるか？」

小課題1(2点)

- $N, Q \leq 3000$

- N, Qが小さい

小課題1解法

- N, Qが小さいので愚直に各クエリ・各学生に対して合否判定をしても間に合う
- 判定は$O(1)$でできるので$O(NQ)$
- 計2点

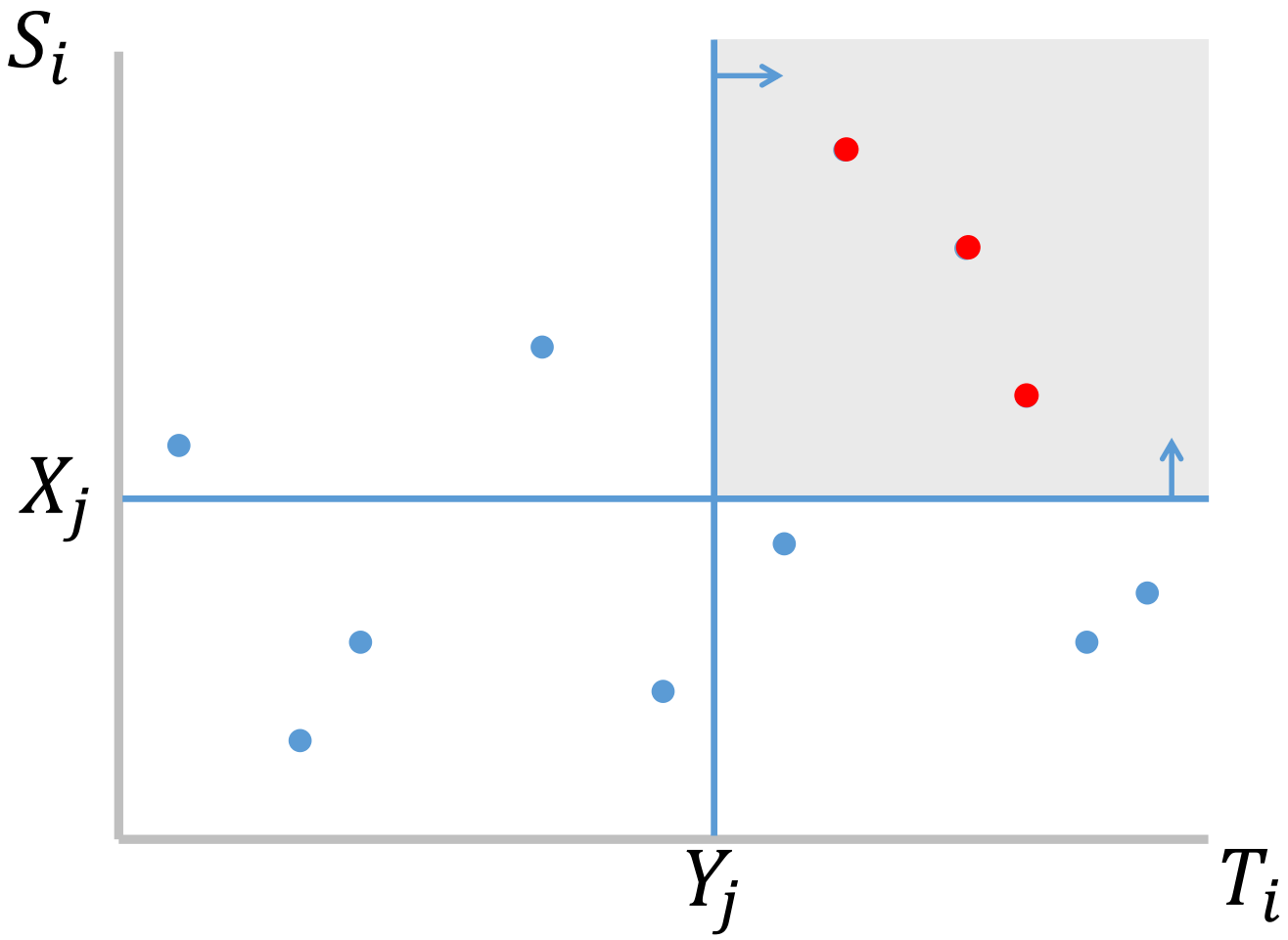
小課題2(20点)

- $N, Q \leq 10^5$
- $S_j, T_j, X_j, Y_j \leq 10^5$
- $Z_j = 0$

- 合計点は合否に影響しない(無視できる)
  - 全員合計点は0点以上のため

- 数学がX_j点以上、情報がY_j点以上の人数を数えたい

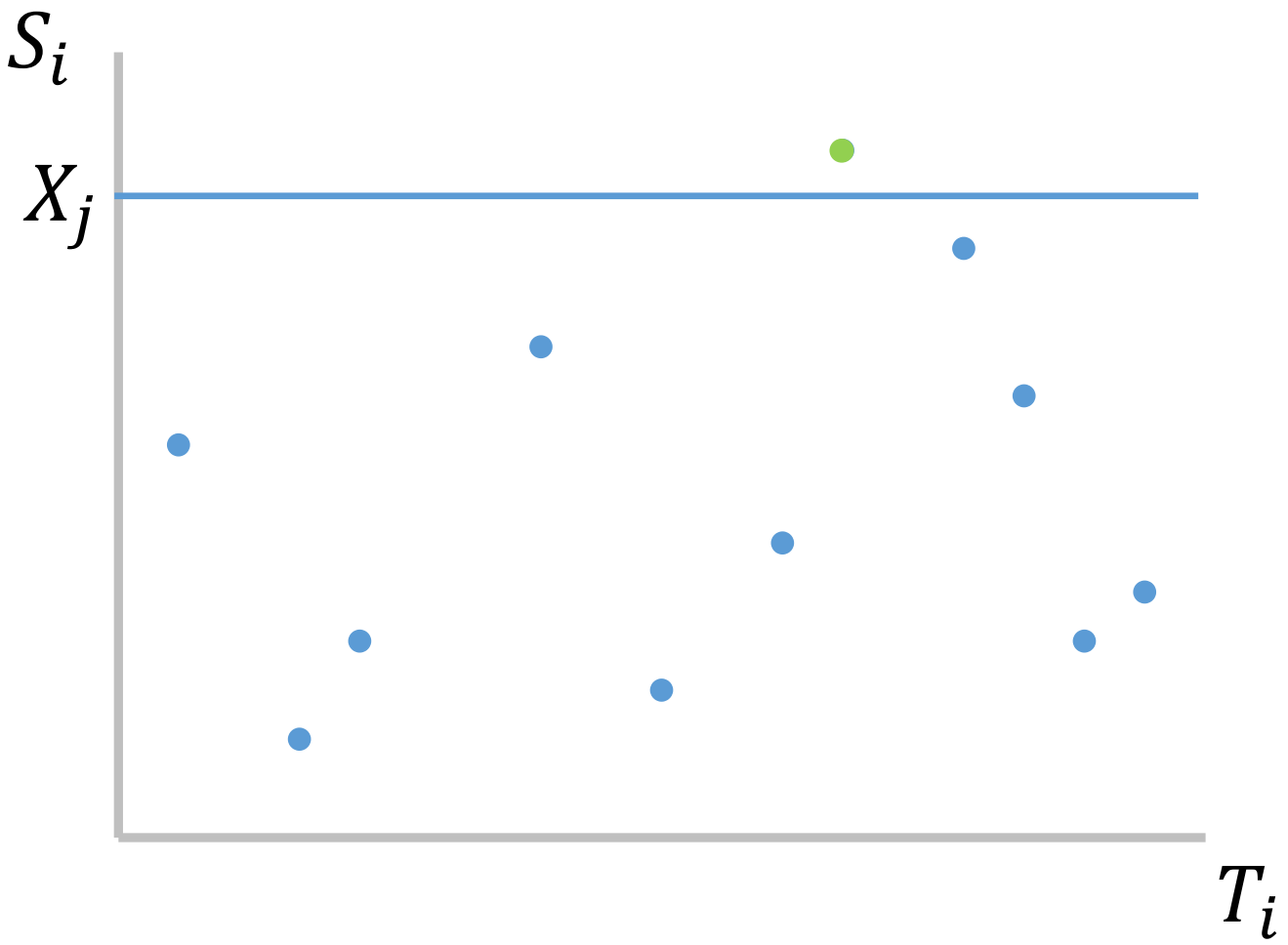
イメージ



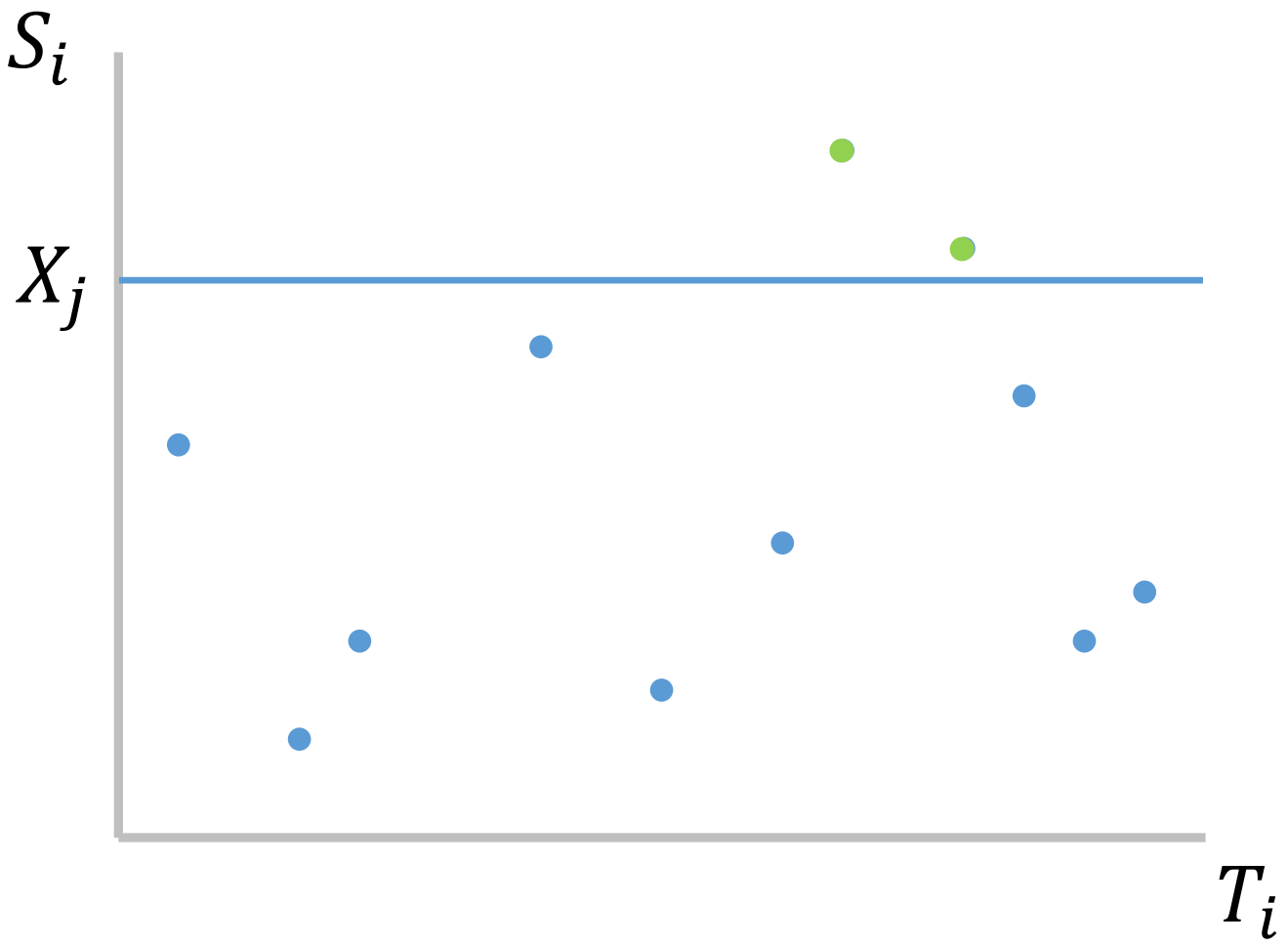
小課題2(20点)

- クエリをx_jの大きい順にソートして、数学の合格基準をだんだん下げていくことを考える

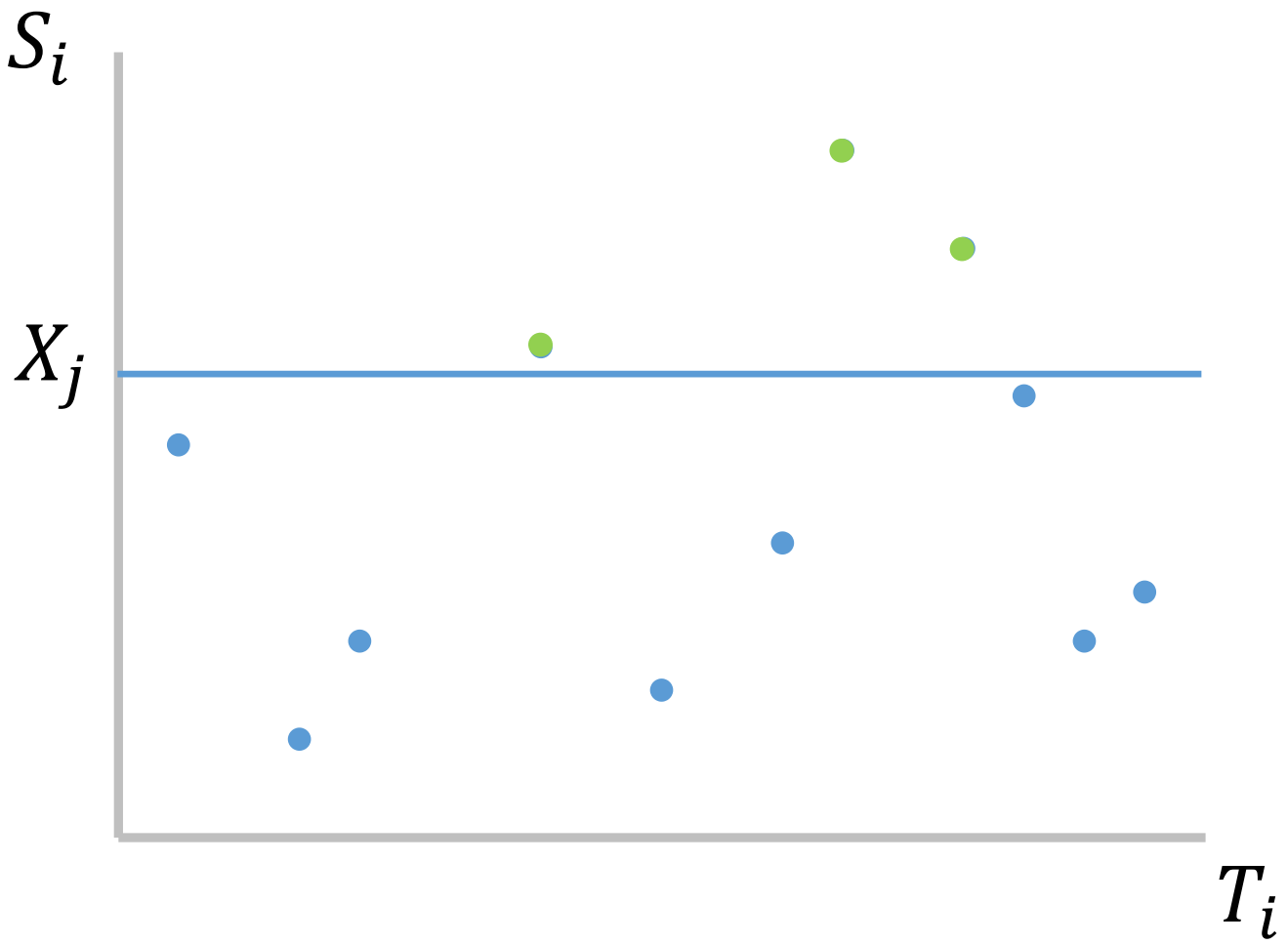
イメージ



イメージ



イメージ



小課題2(20点)

- 新たに数学の合格基準を満たした学生について、T_iの値を記録する

→数列$b_1, b_2,\cdots$(初期値はすべて0)に対し、b_{T_i}に1を足す

- Y_j以上の値がいくつあるかを求める

→$b_{Y_j}, b_{Y_j+1},\cdots$の和を求める

の二つの操作ができればよい

- BIT(Binary Indexed Tree)を使って高速に解ける

小課題2解法

- X_jの大きい順にクエリをソート
- 数学がX_j点を超えた学生から順にBITのT_iのところに1を足していく
- クエリではBITで$[Y_j, \infty)$のところの和を求めて答えればよい
- 計算量は$O((N+Q)\log \max(\text{得点}))$
- 計22点

小課題3(21点)

- $N, Q \leq 10^5$
- $S_j, T_j, X_j, Y_j \leq 10^5$
- $Z_j \leq 2 \times 10^5$

- 合計点も合否に影響してくる

小課題3(21点)

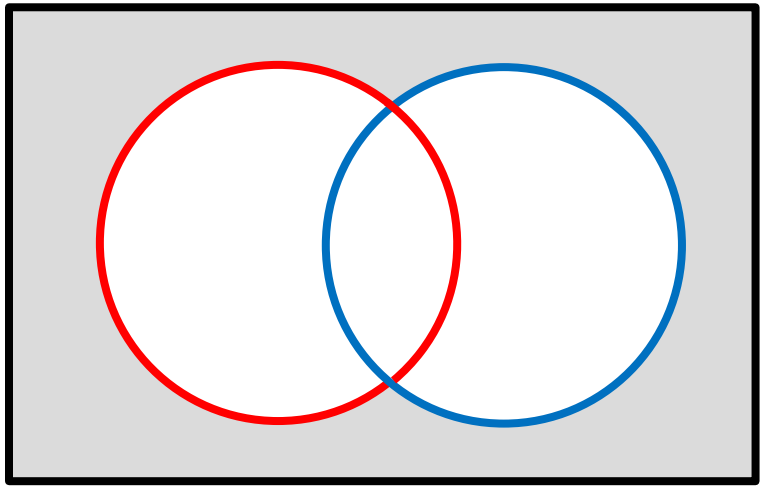
- $X_j + Y_j \geq Z_j$のときは合計点の制約を無視できる

→そのようなクエリには小課題2の解法をそのまま使うことにする

- 問題は$X_j + Y_j < Z_j$のとき

- 合計点がZ_j点以上の学生のうち、数学と情報も合格基準を満たしている人数の数え方を考える

少し考えてみる



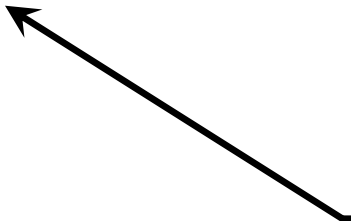
合計点の合格基準を満たしている学生のうち

- ○内:数学の合格基準を満たしている
- ○内:情報の合格基準を満たしている
- ■:数学・情報ともに合格基準を満たしていない

- ○・○の人数は小課題2のようにソートとBITで求まりそう
  - 合計点の合格基準を下げていき、新たに合格基準を満たした学生の得点をBITに記録していく
- ■の人数がわかれば両方合格基準を満たしている人数も求まって幸せ

よく考えてみる

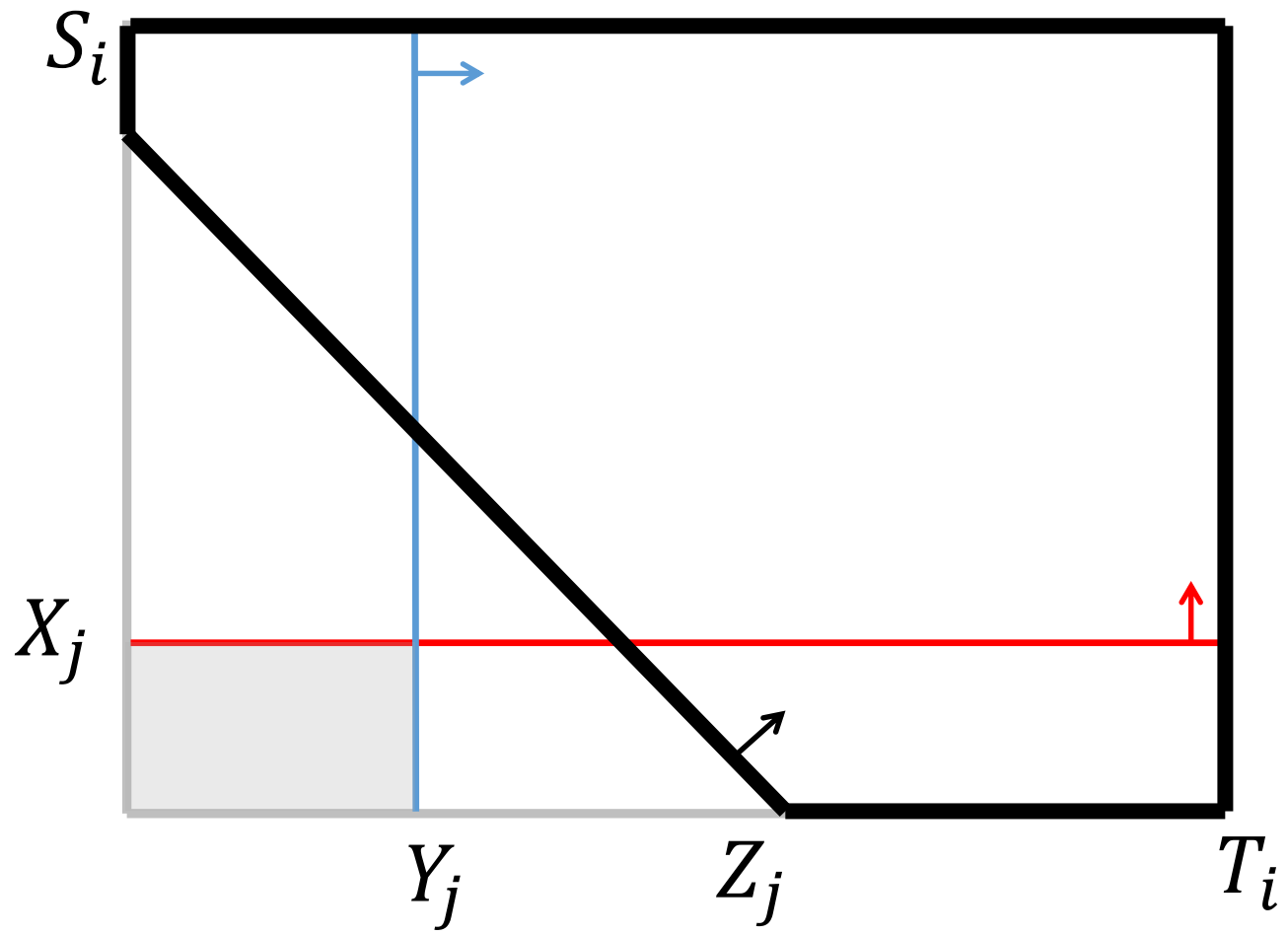
- 数学がX_j点未満、情報がY_j点未満なら合計点は$X_j + Y_j$点未満
  - $S_i < X_j, T_i < Y_j \Rightarrow S_i + T_i < X_j + Y_j$



- ここで$X_j + Y_j < Z_j$のときを考えているのでこのような学生はそもそも合計点の基準を満たしていない

→ ■ の人数は0人！

イメージ



■ は黒枠(合計点の合格基準を満たしている領域)内に含まれていない

小課題3解法

- $X_j + Y_j \geq Z_j$であるクエリは小課題2と同じ
- $X_j + Y_j < Z_j$であるクエリはZ_jの大きい順にソート
- 2つのBITを用意
  - 数学の得点を記録する用
  - 情報の得点を記録する用
- 合計点がZ_j以上の学生について、2つのBITそれぞれに得点を記録していく

小課題3解法

- クエリでは
(数学合格者数) + (情報合格者数) − (それまでに追加した学生数)
を答えればよい

- 計算量は小課題2解法と同じ

- 計43点

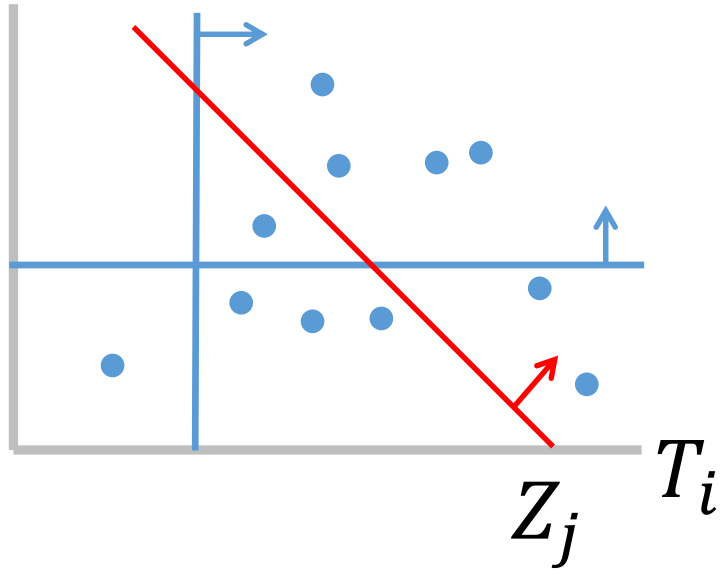
小課題4(57点)

- 点数の値が大きい
- 点数について座標圧縮をすれば実質小課題3と同じ
- 計算量$O((N+Q)\log N)$
- 計100点

別解:二次元セグメント木

- 合計点の合格基準を下げていくとき、「数学がX_j点以上、情報がY_j点以上である学生の数」を求められればよい
  - 下図赤線を左に動かしていく

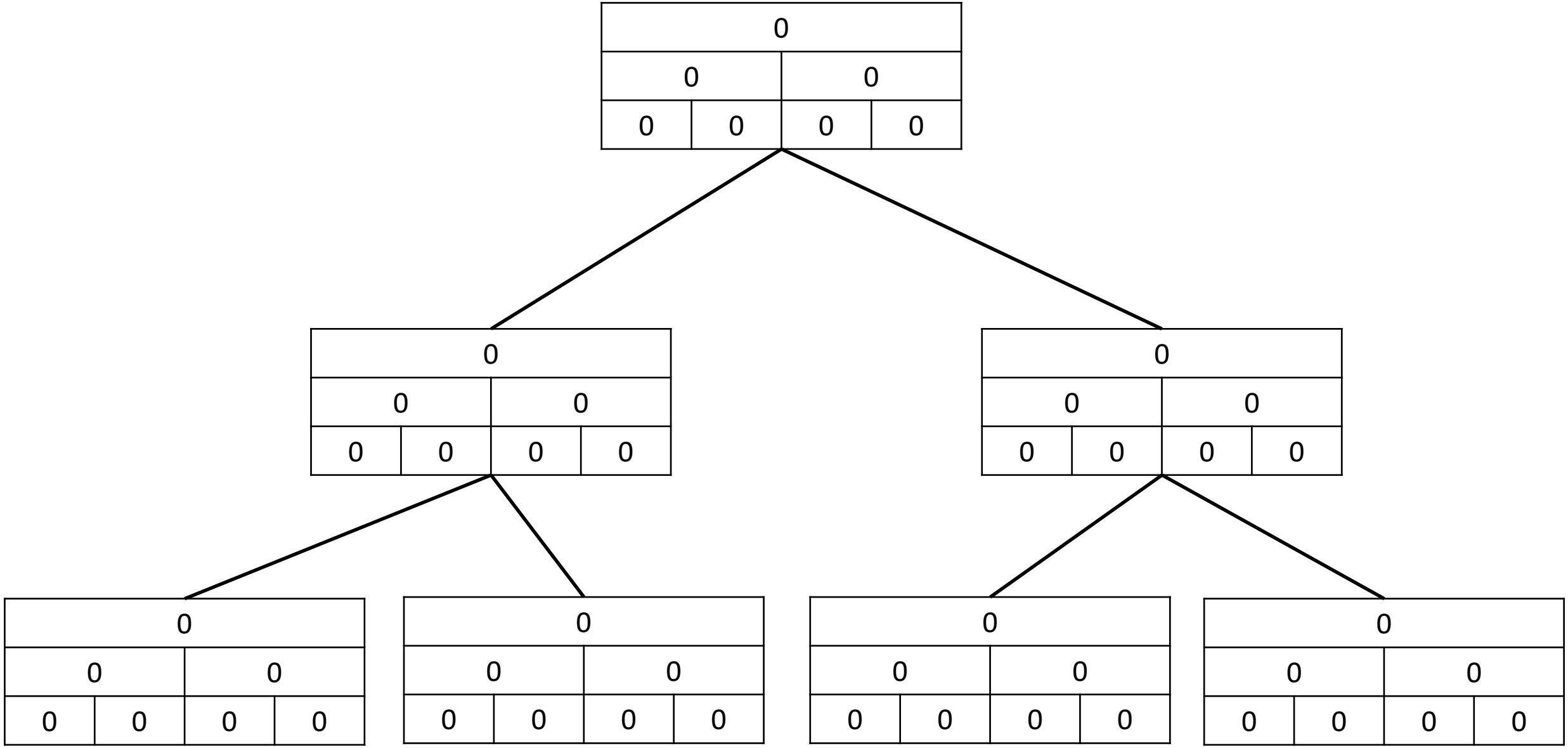
- 次の2つのクエリが高速に処理できればよい:
  - 座標平面に点を追加
  - 矩形範囲の点の個数を求める

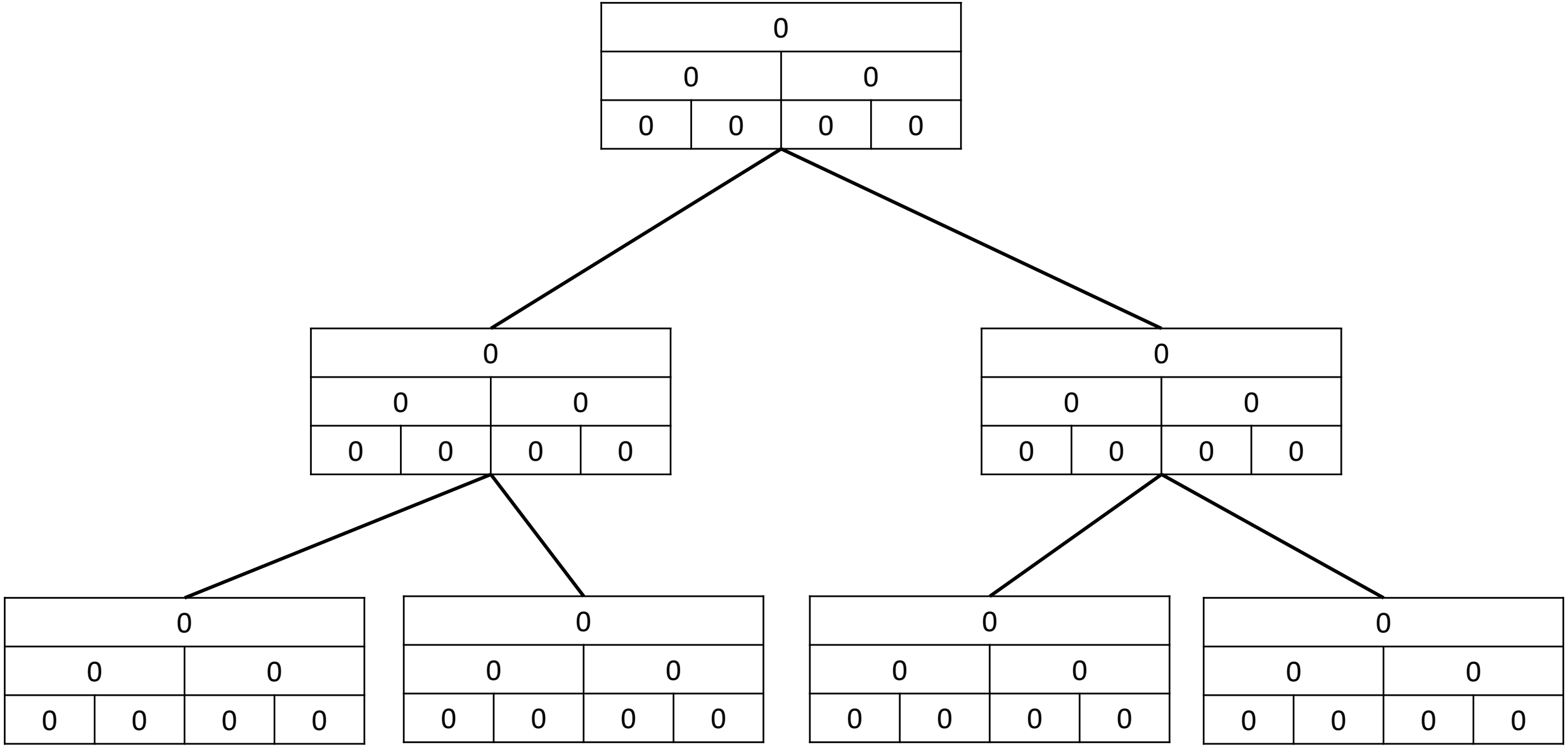


→セグメント木のノードに適切なデータ構造をのせればできる

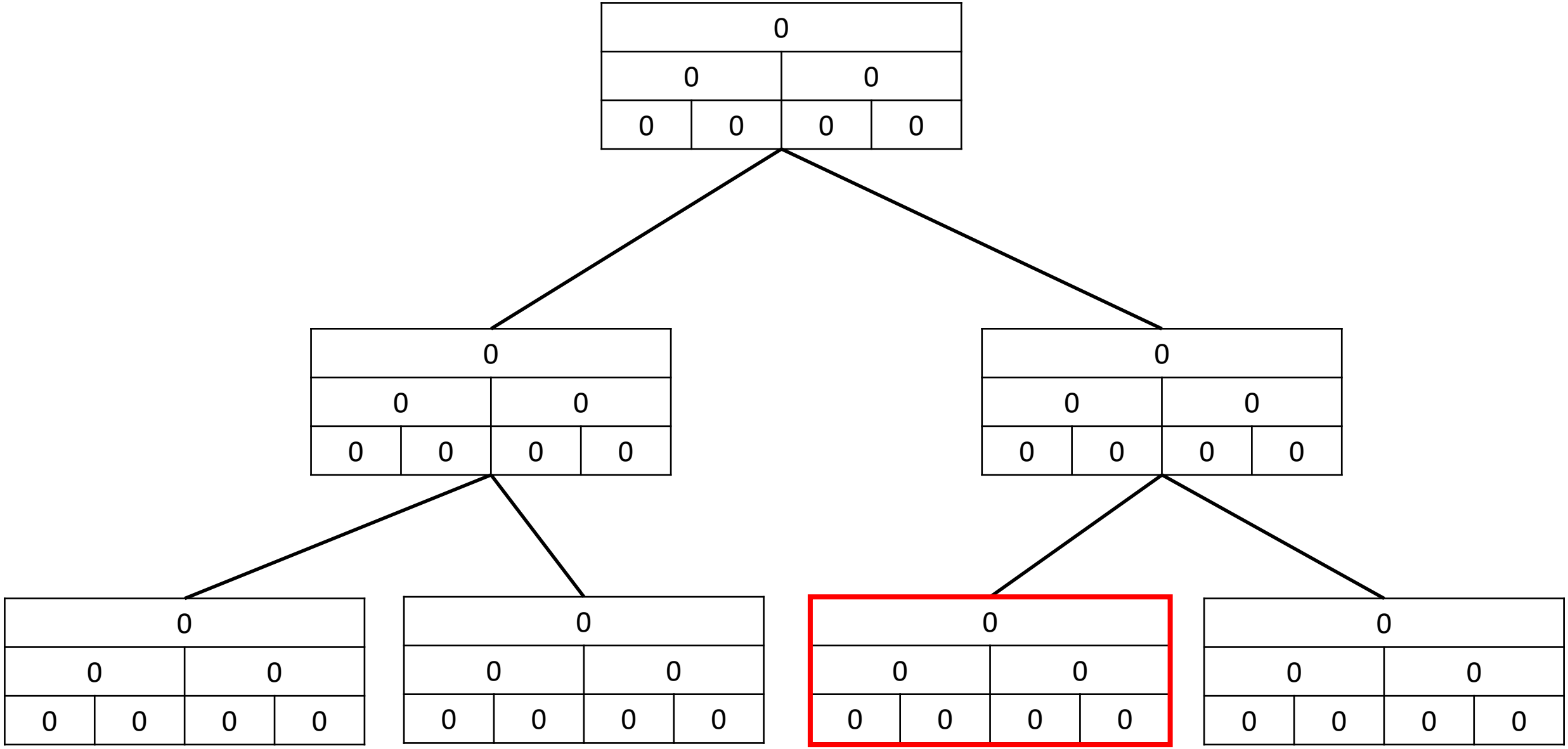
方法1:セグメント木 on セグメント木

- セグ木のそれぞれのノードがセグ木をもつ
- 全体のセグ木上で、横の範囲に対応するノードまで潜る
- そのノードのセグ木上で、縦の範囲に対応するノードに対し操作する
- 子の変更を親に伝播する

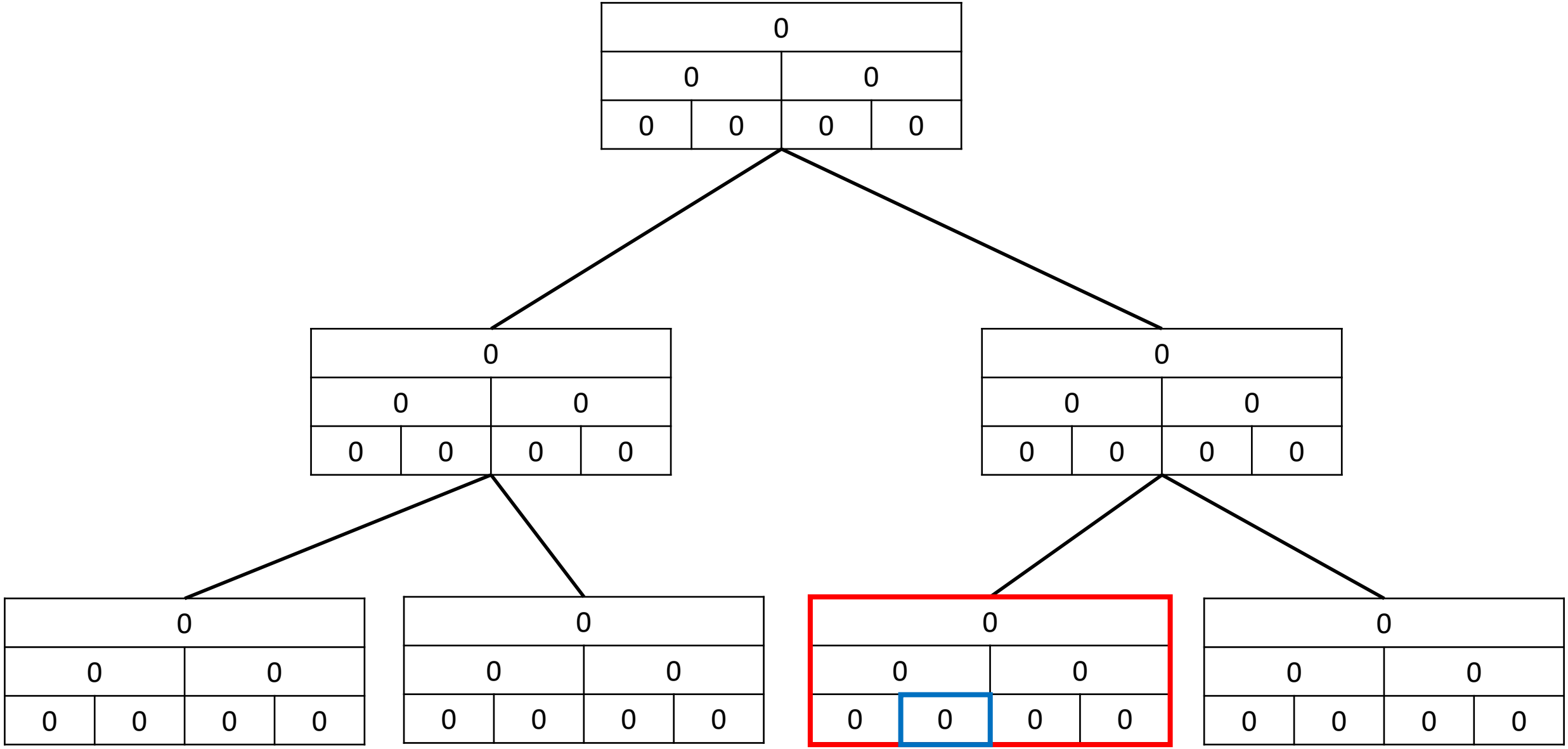

0
0 0
0 0 0 0
0
0 0
0 0 0 0
0
0 0
0 0 0 0
0
0 0
0 0 0 0
0
0 0
0 0 0 0
0
0 0
0 0 0 0
0
0 0
0 0 0 0



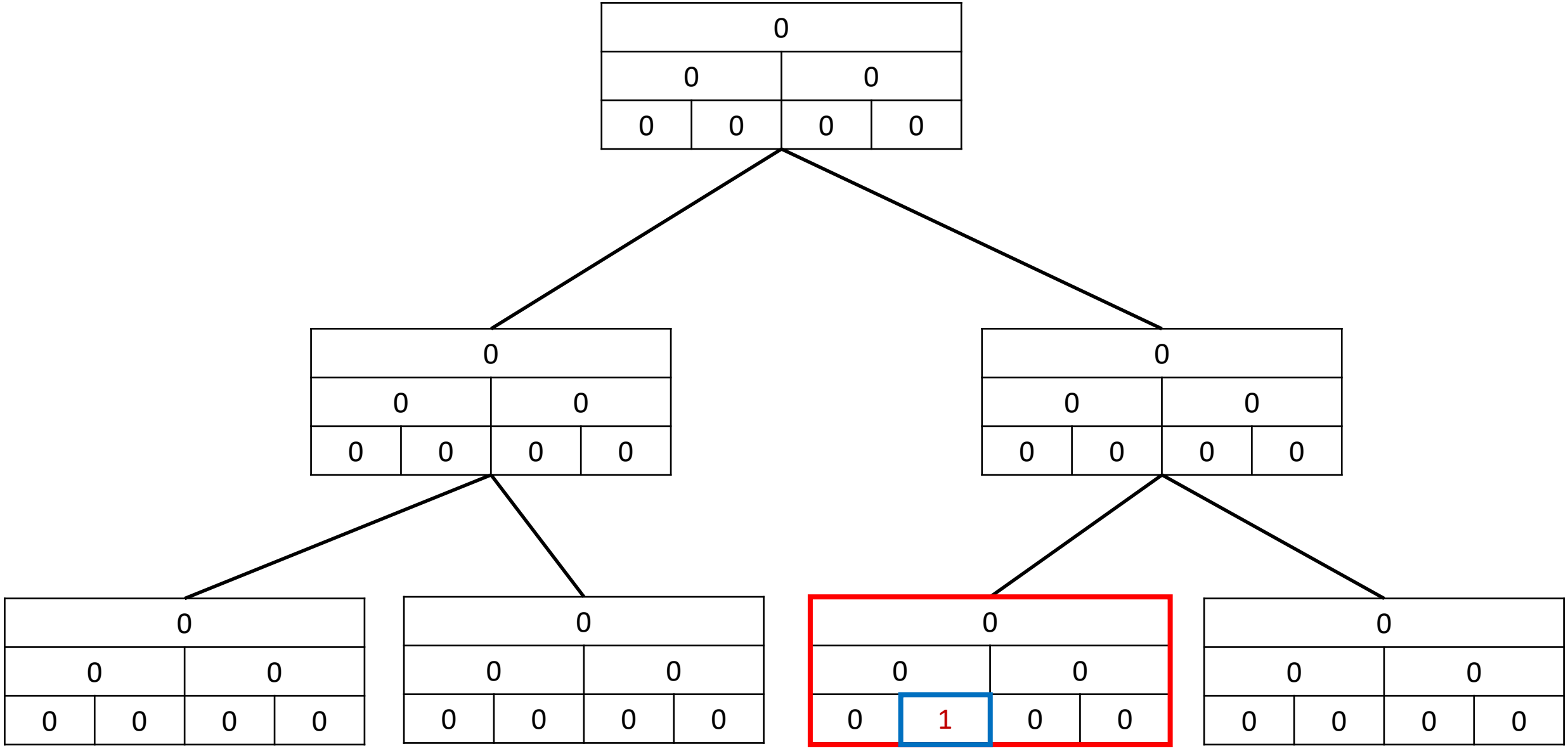
座標(2,1)に点を追加



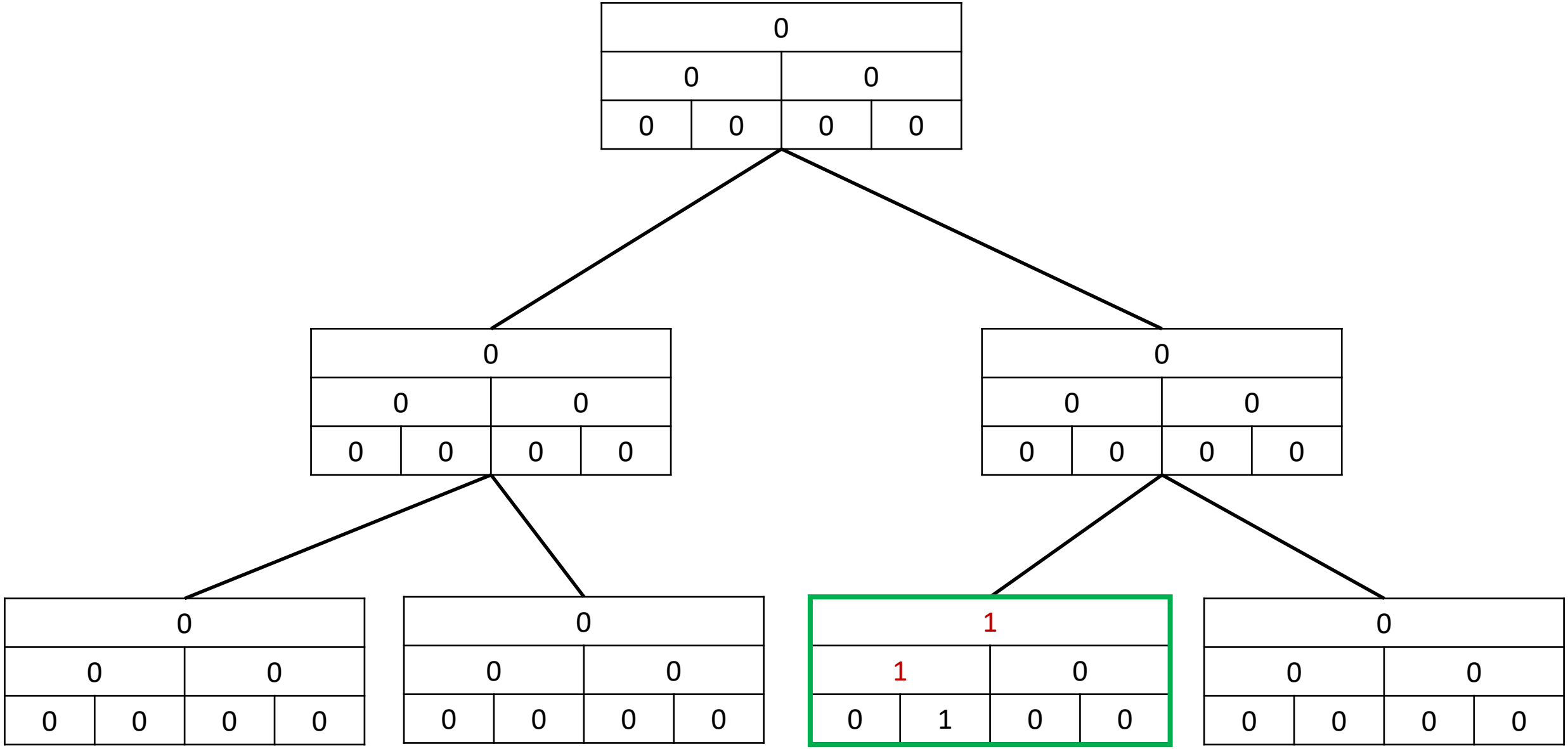
座標(2,1)に点を追加



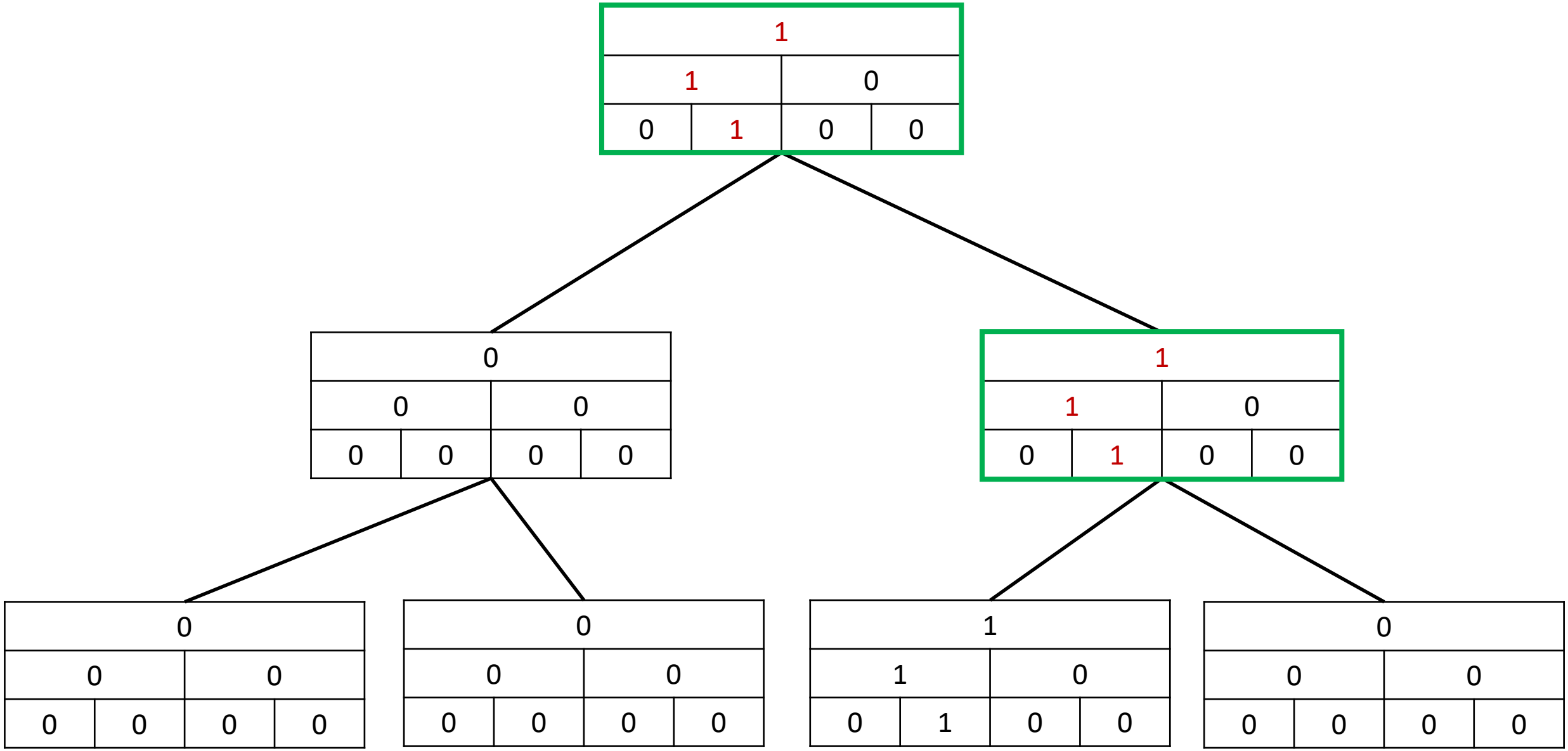
座標(2,1)に点を追加



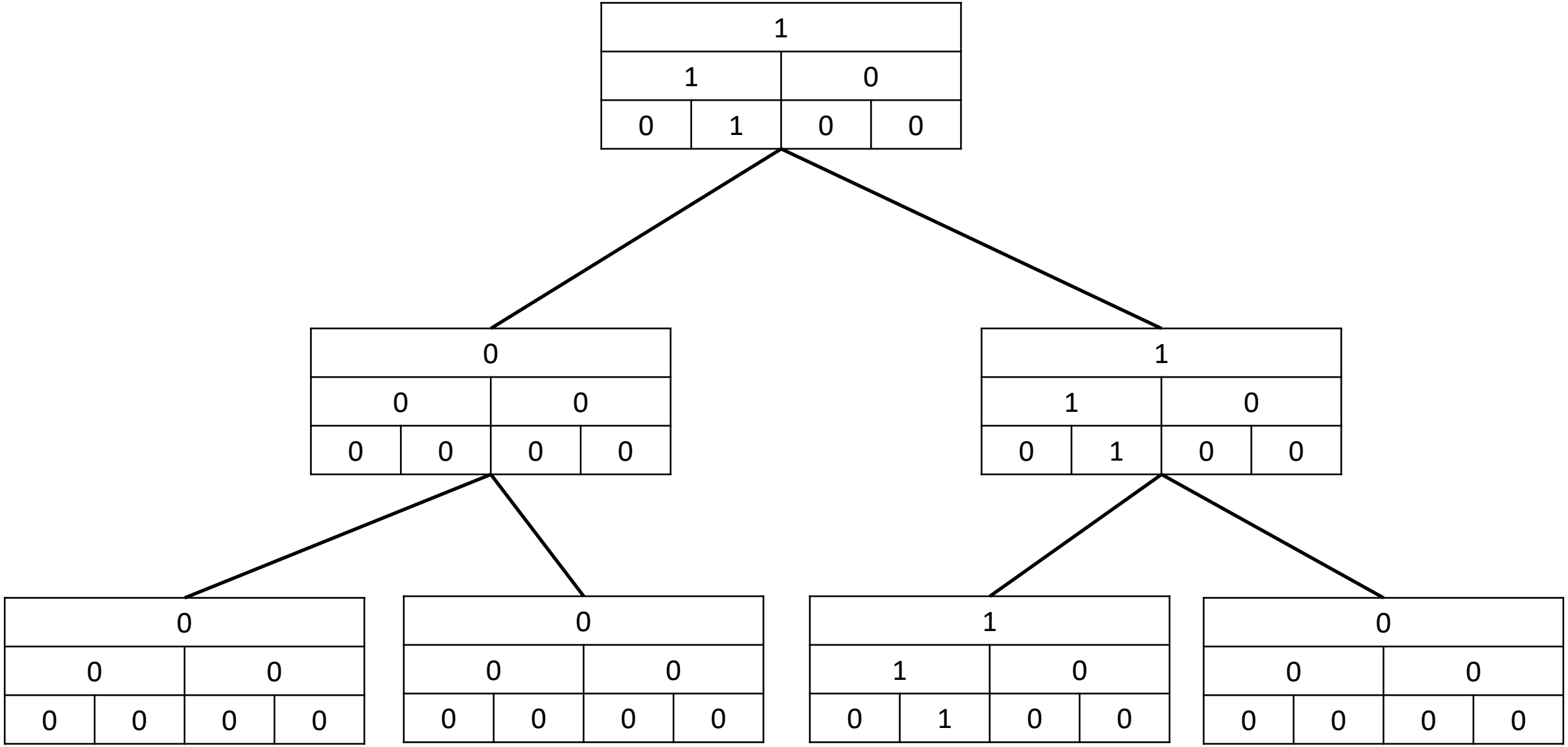
座標(2,1)に点を追加



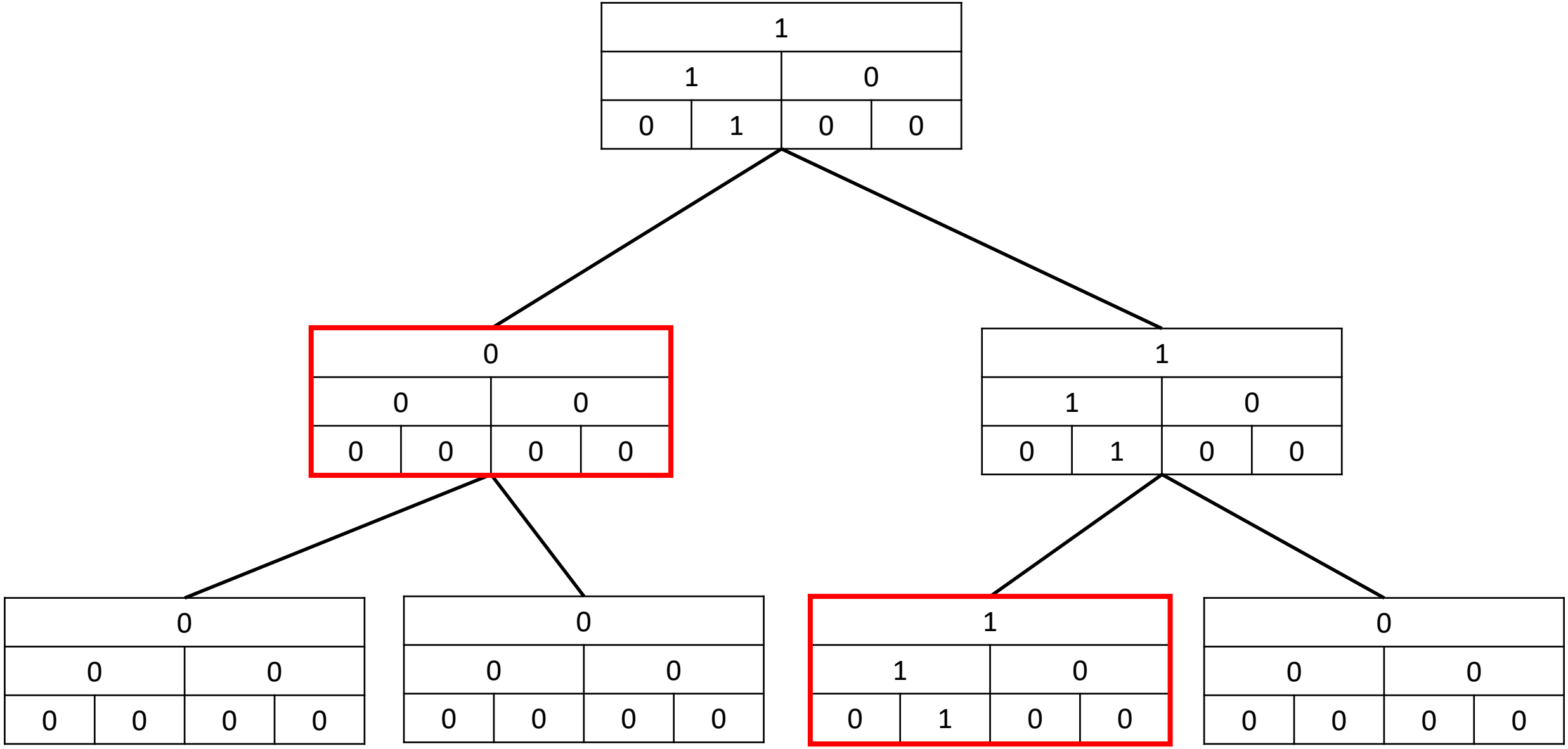
座標(2,1)に点を追加


1
1
0
0
1
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
1
1
0
0
1
0
0
0
0
0
0
0
0
0
0
0
1
1
0
0
1
0
0
0
0
0
0
0
0
0
0
0

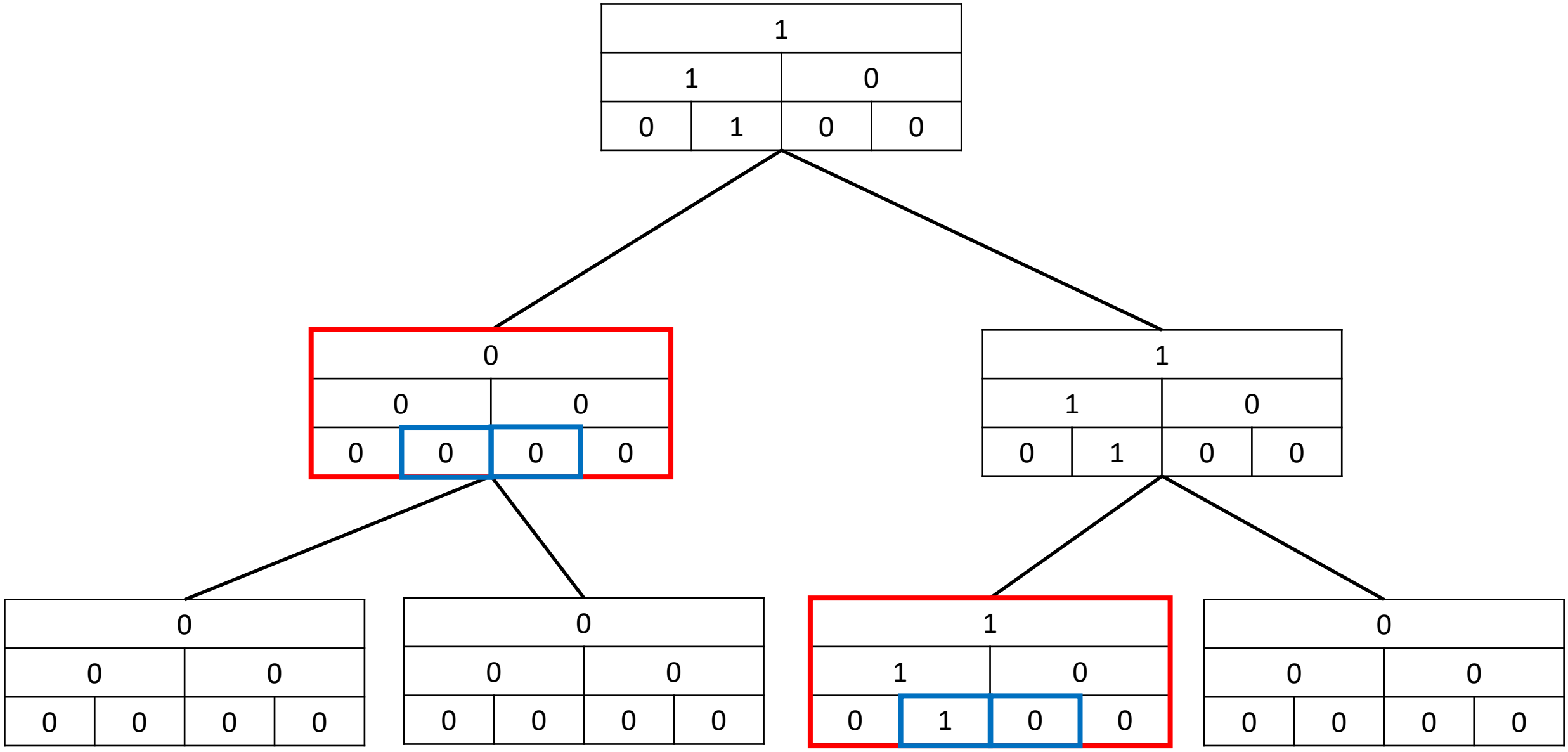
座標(2,1)に点を追加



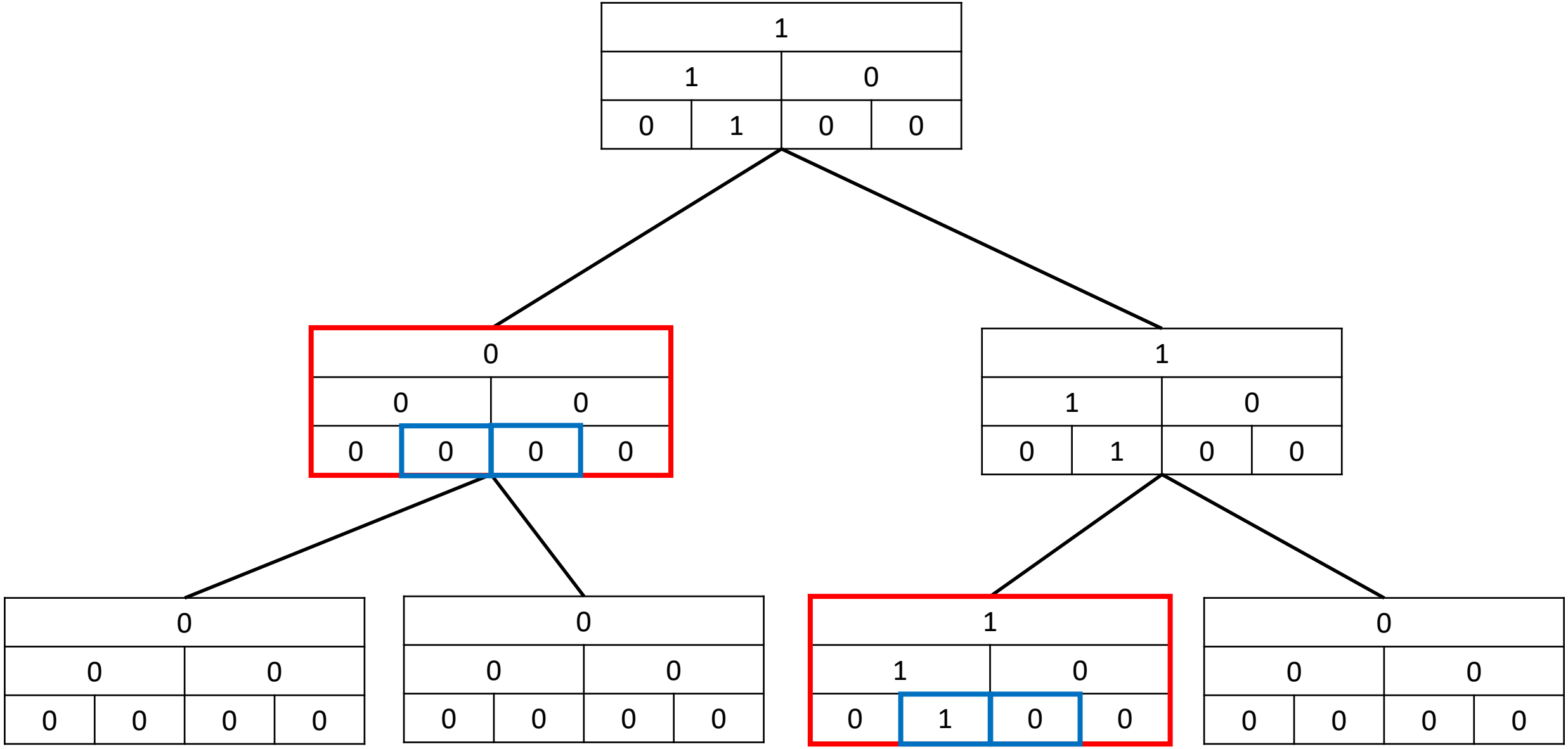
範囲[0,3)x[1,3)内の点の個数を求める



範囲[0,3)x[1,3)内の点の個数を求める



範囲[0,3)x[1,3)内の点の個数を求める



範囲[0,3)x[1,3)内の点の個数を求める→1

方法1:セグメント木 on セグメント木

- 1回の操作では大きなセグ木で$O(\log N)$,それぞれのノードで$O(\log N)$かかるので全体の時間計算量は$O\left((N+Q)\,(\log N)^2\right)$
- ただしそのままでは空間計算量$O(N^2)$でMLEなので工夫する
- クエリの先読みができるので、それぞれのノードのセグメント木であらかじめ使われる値だけ座標圧縮しておけば幸せになれる

方法2:平衡二分木 on セグメント木

- 値の挿入とある値以上の値の数の取得ができる平衡二分木を使う
  - C++ STLのsetでは後者ができない
- g++拡張のtreeを使うと楽
  - 参照:http://hogloid.hatenablog.com/entry/2014/09/23/132440
- 計算量$O((N+Q)\log^2)$くらい
  - ただし定数倍は重いので注意

点数分布