Construction / 建設計画

2013/03/21

JOI Spring Camp day2

Shogo Murai

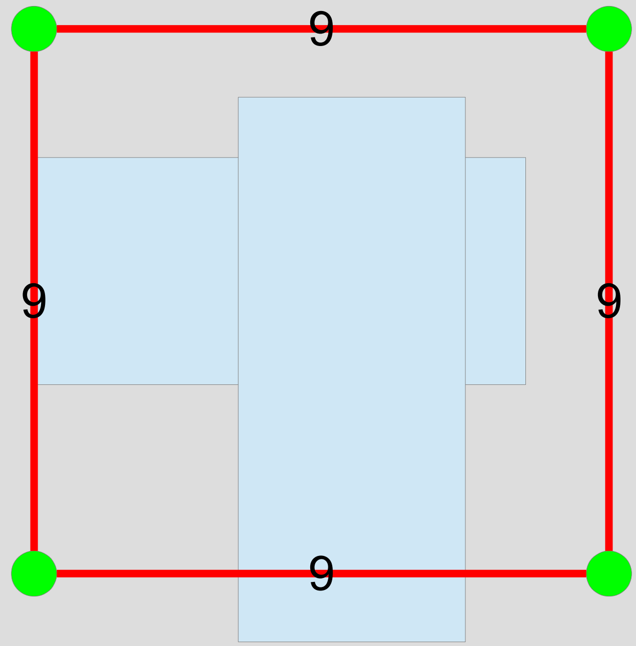
問題の概要

- N 個の町があり、国際空港と道路を建設する
- 道路は、町同士を結ぶ x , y 軸に平行な線分
- 長さ L の道路を建設するのにはコストが L かかる
- M 個の長方形の領域には道路を建設できない
- どの町からも道路を通って国際空港のある町に行けなくてはならない

- 建設業者が C 社あり、それぞれ国際空港の建設コストと最大建設数が決まっている
- 交通網の整備にかかる各社のコストを求める

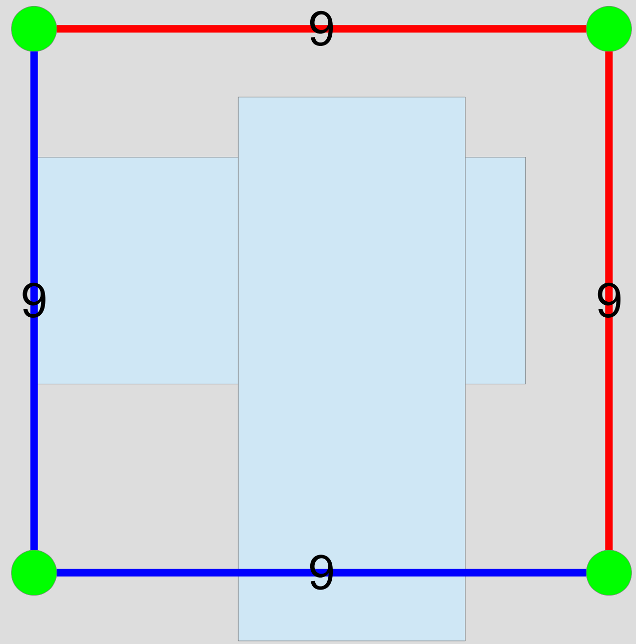
サンプルの解説 (1)

- ●が町、□が道路を敷設できない場所
- 赤線は引けるかもしれない道路
- 数字は道路の長さ
- 斜めには道路を敷設できない



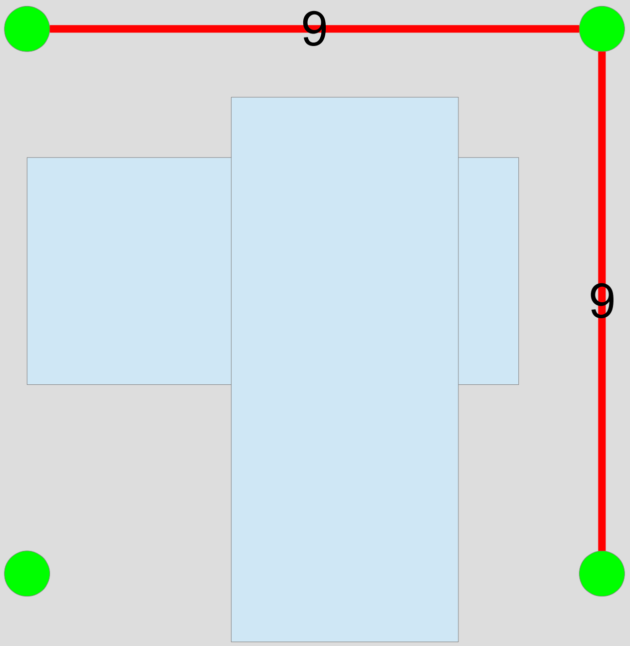
サンプルの解説 (2)

- 青色の道路はと交わる
  → 建設できない
- 長方形の外周部と交わってもだめ



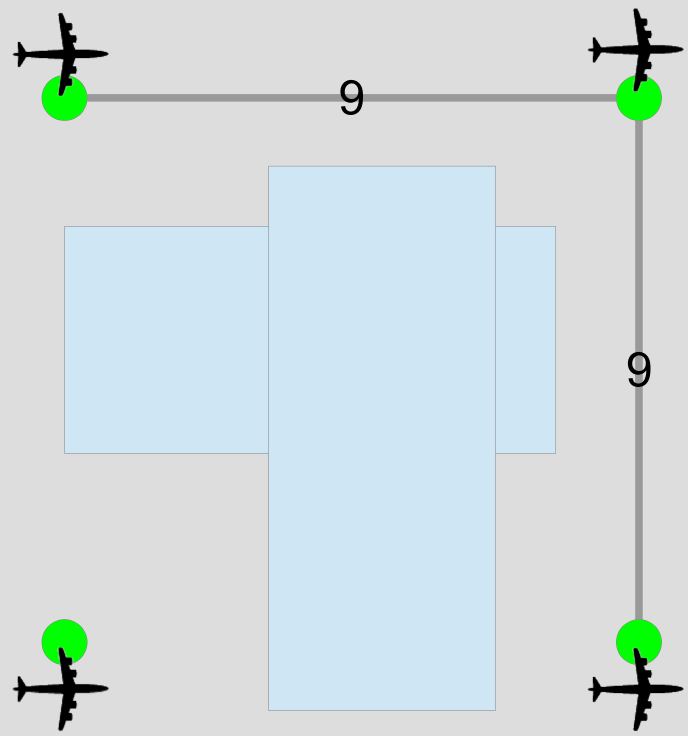
サンプルの解説 (3)

- 使える道路はこんな感じ



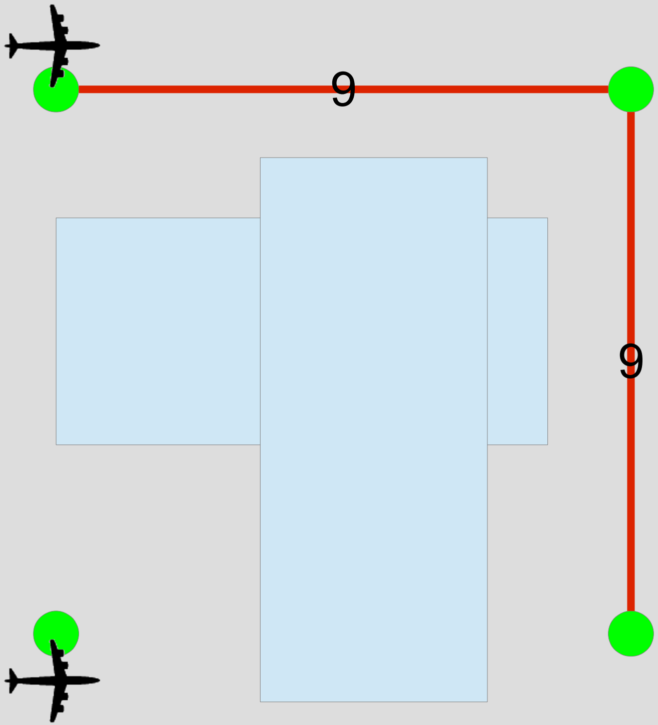
サンプルの解説 (4)

- 建設会社 1
  - 空港のコスト: 7
  - 建設できる数: 4
- 右のように建設するのが最適(: 空港)
- コストは 28



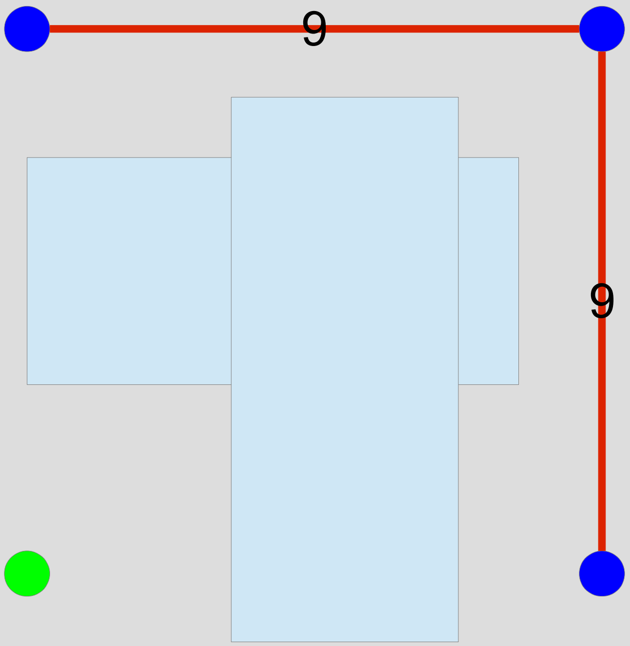
サンプルの解説 (5)

- 建設会社 2
  - 空港のコスト: 10
  - 建設できる数: 3
- 例えば、右のように建設するのが最適
- コストは 38



サンプルの解説 (6)

- 建設会社 3
  - 空港のコスト: 1
  - 建設できる数: 1
- どうやったって空港 2 つは必要
- 無理、答えは -1

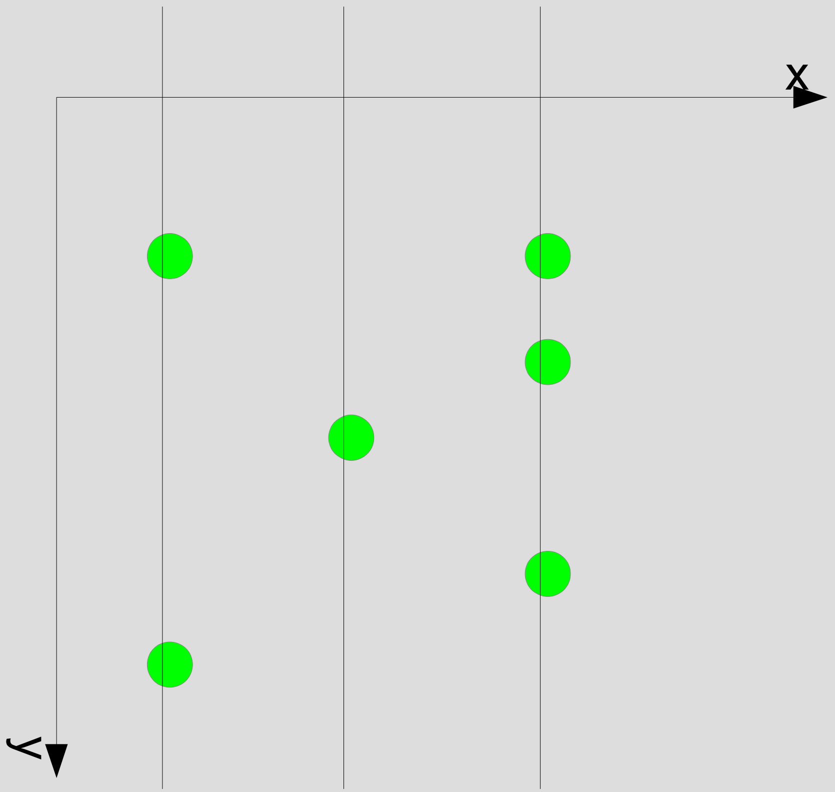


考えること

1. 建設できる道路はどれ？
2. 道路はどの長方形とも交わらない？
3. 使える道路網がわかったとき、道路のよい建設方法は？
4. 各社の建設コストは？

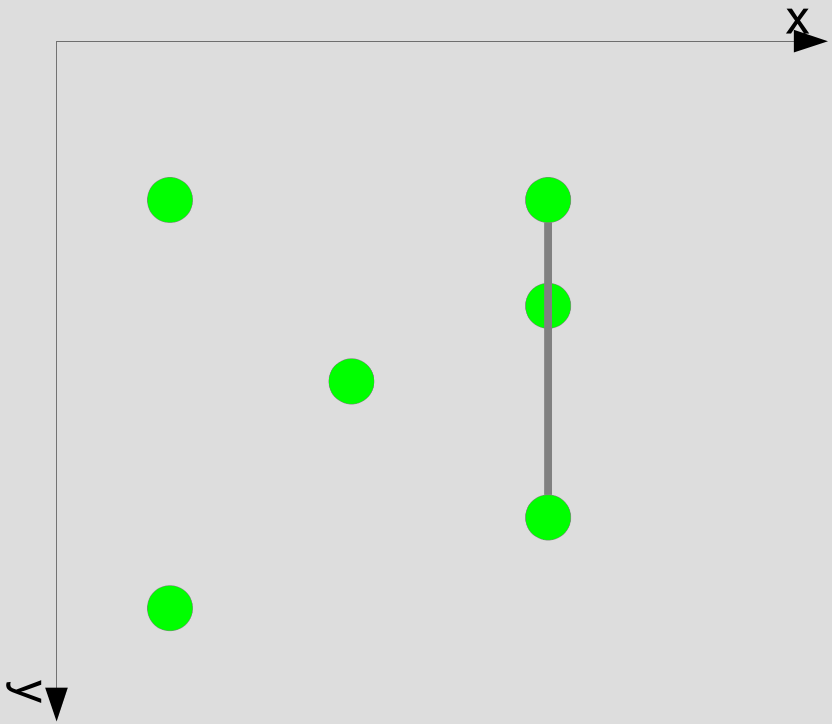
1. 建設できる道路(1)

- とりあえず y 軸に平行なものだけ考える
- すると、道路が結ぶ町の x 座標は等しい



1. 建設できる道路(2)

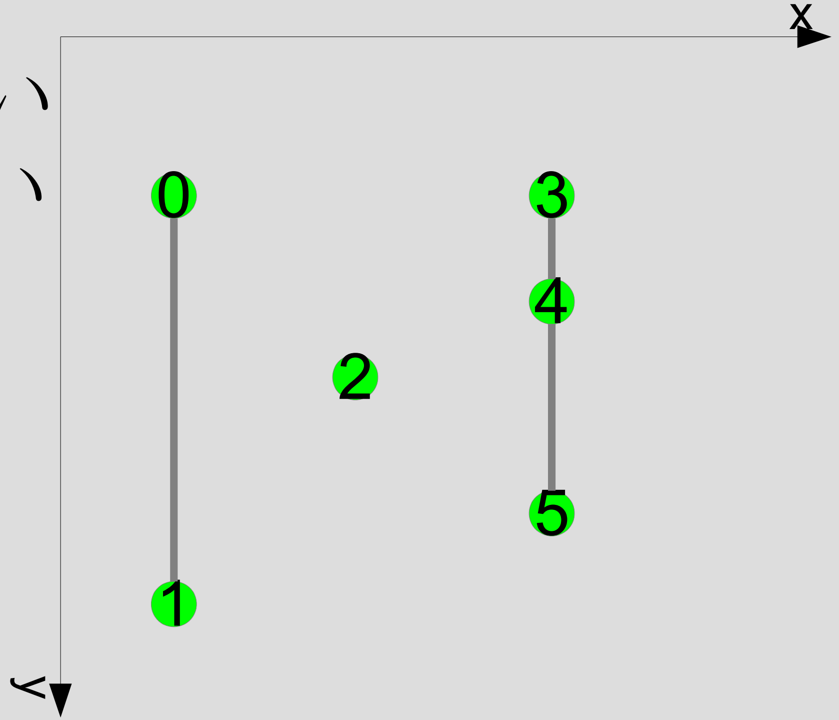
- 間に他の町があるのに飛び越えて道路を建設するのは無意味
- 間に町がない道路だけ考えれば良い



1. 建設できる道路(3)

- (x,y) を pair としてソート
- ソート列上で隣り合っていない所には道路はない
- N-1 箇所だけ見ればよい
- 計算量: O(NlogN)
- 道路は高々 N-1 本

- 2. をナイーブにやると O(NM)

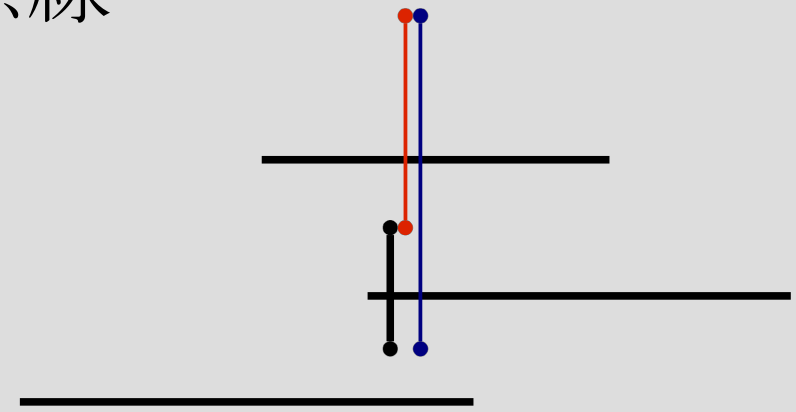


2. 長方形との交差判定(1)

- y 軸に平行な線分と長方形が交わる条件は？
- 長方形の x 軸に平行な 2 辺だけ見ればよい（理由）
- y 軸に平行な辺と共通部分を持つとき、問題文の条件から端点と共通部分を持つ
- よってこのとき x 軸に平行な辺とも交わる

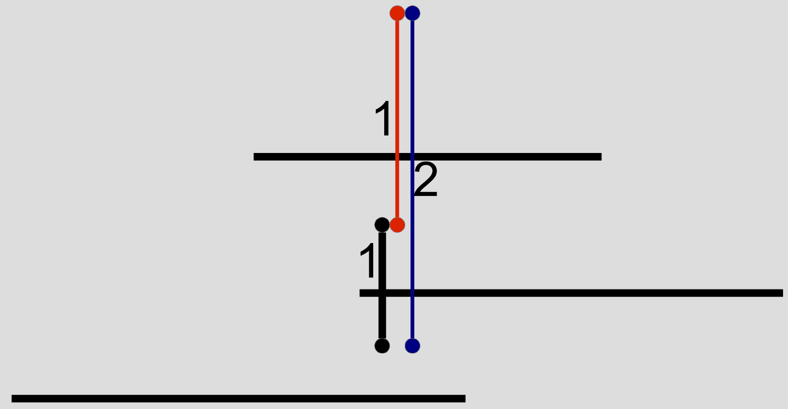
2. 長方形との交差判定(2)

- 線分と線分の交差判定を考えればよい
- y 軸に平行な線分が x 軸に平行な線分たちとそれぞれ何回交わるかわかればよい
- 図の、(赤線が交わる回数)−(図の青線が交わる回数) で黒線が交わる回数は求まる



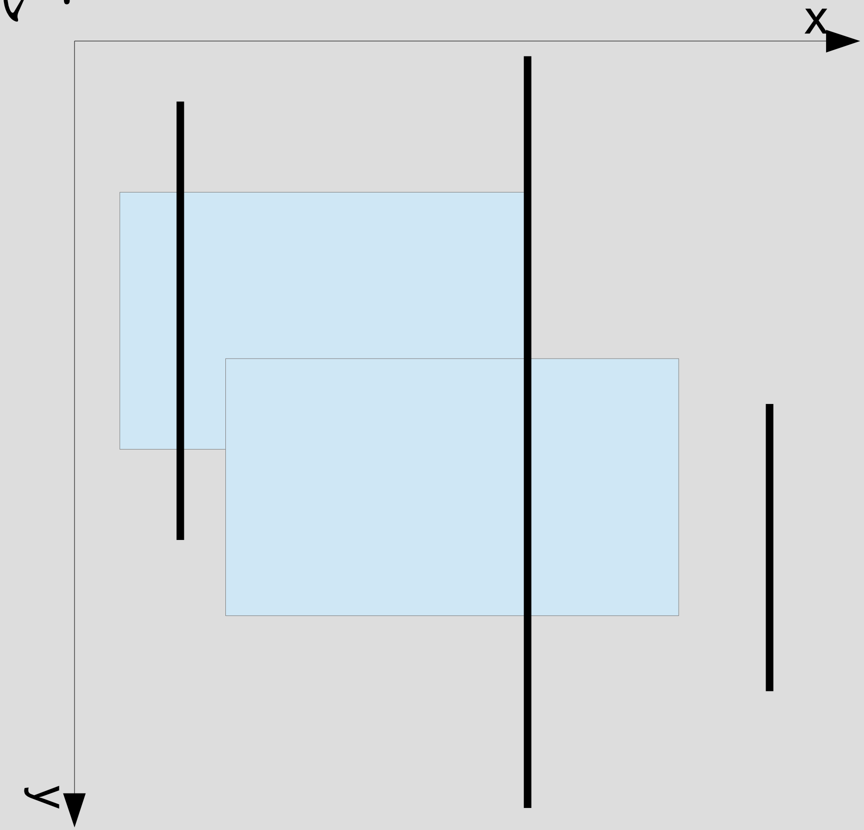
2. 長方形との交差判定(3)

- y 座標の小さいほうから見ていく(走査)
- 走査の過程で赤線、青線の終端が出てきたら、その線のある x 座標に対し、今までにそこをまたいだ線分の数を求める
- その方法は後述



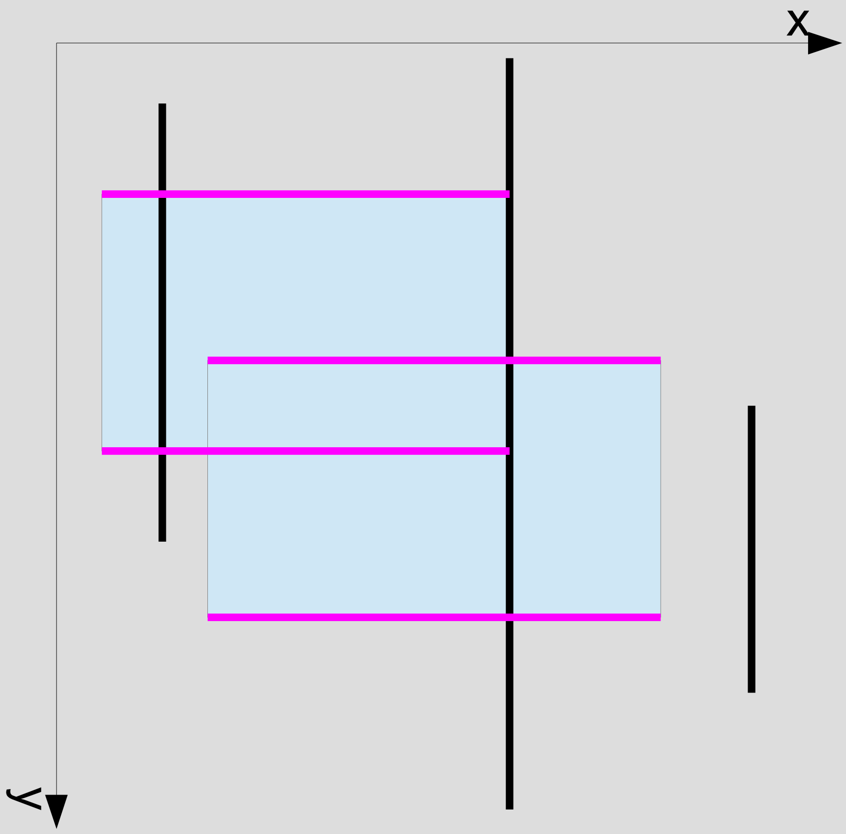
2. 長方形との交差判定(4)

これだけでは分かりにくいので具体例を挙げて説明します(右図)



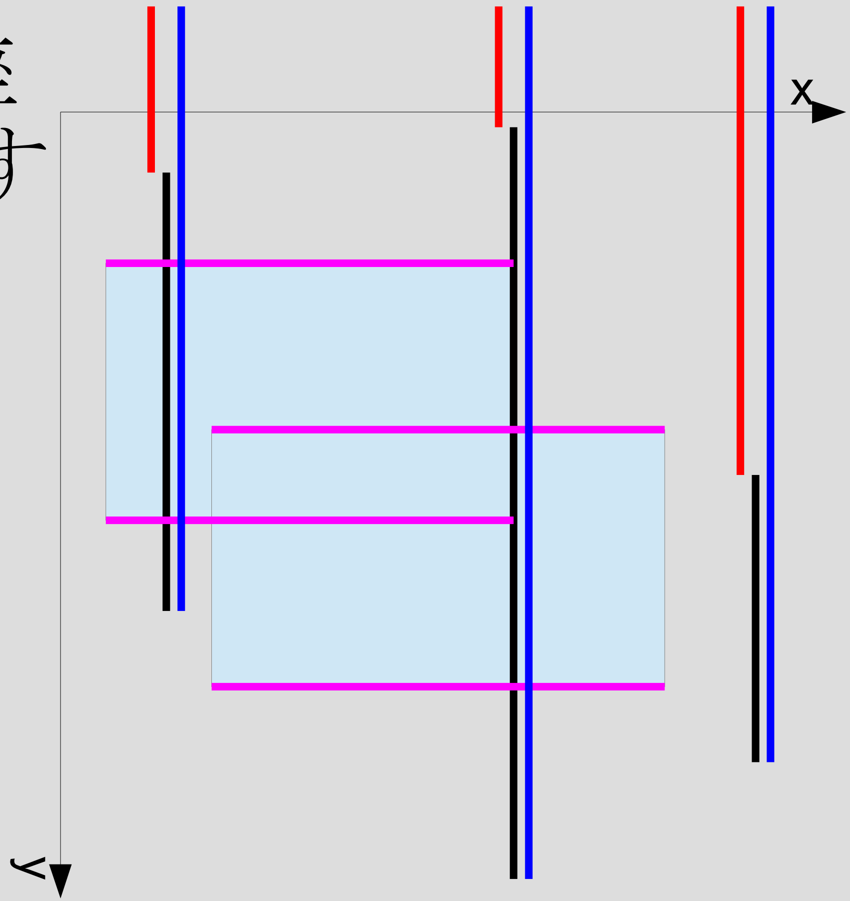
2. 長方形との交差判定(5)

- 長方形の x 軸に平行な辺だけを取り出す



2. 長方形との交差判定(6)

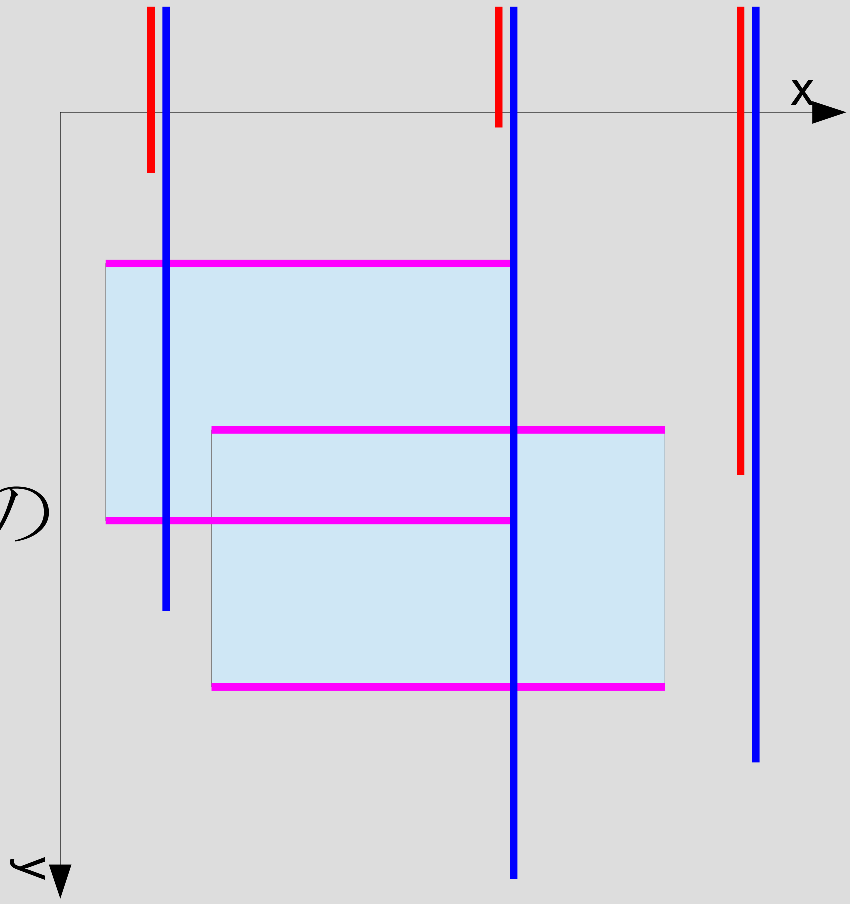
- 各道路に対し、同じ x 座標に青線と赤線を生やす
- 青線、赤線は y=-inf から生えている



2. 長方形との交差判定(7)

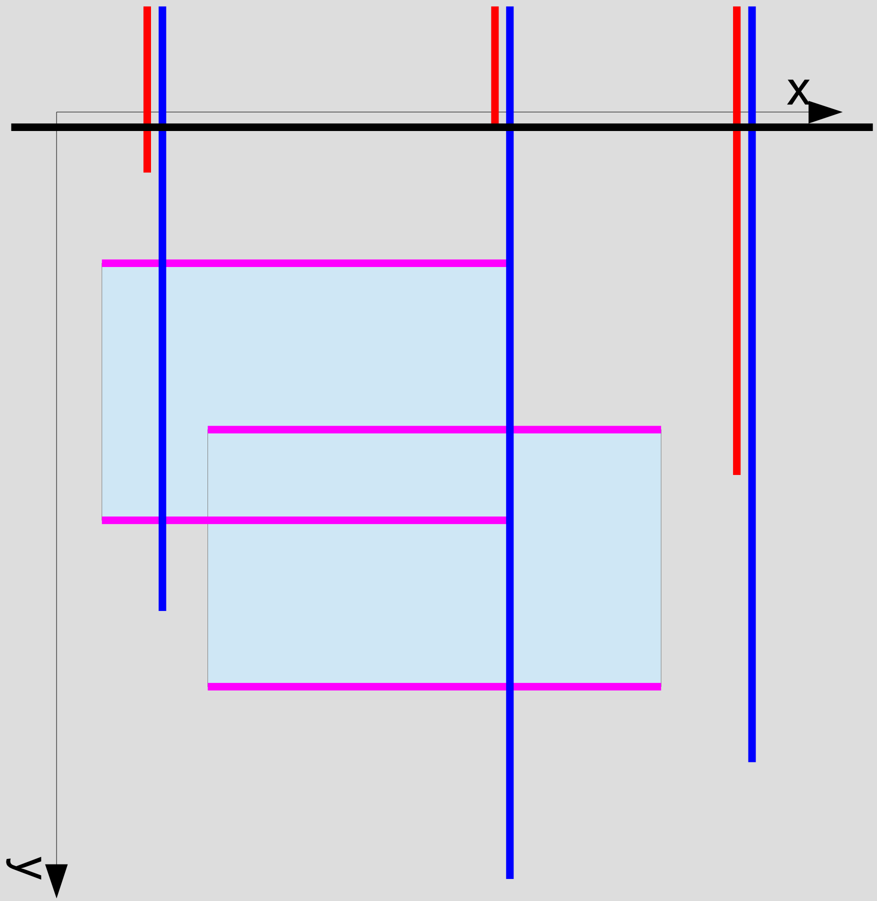
- y 座標の小さいほうから走査線を動かし、
  - 赤線、青線の下端
  - x 軸に平行な線分

  が出てくるたびに処理
- 実装上は、下端や線分の y 座標でソートして処理



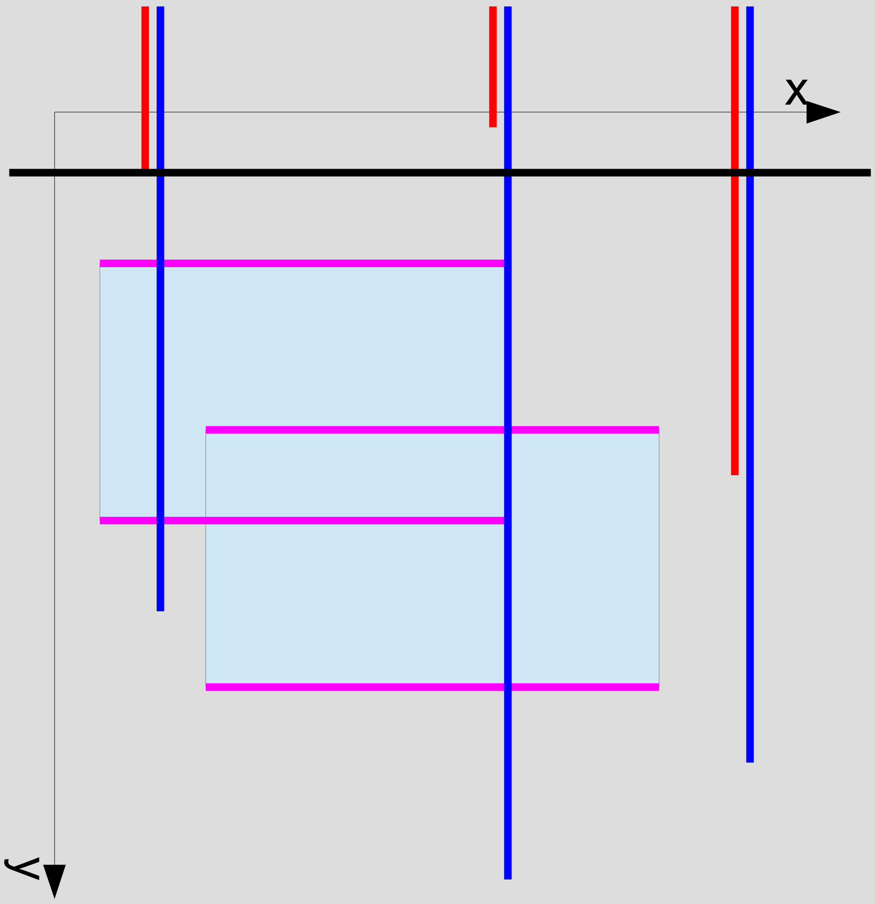
2. 長方形との交差判定(8)

- 赤線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 0



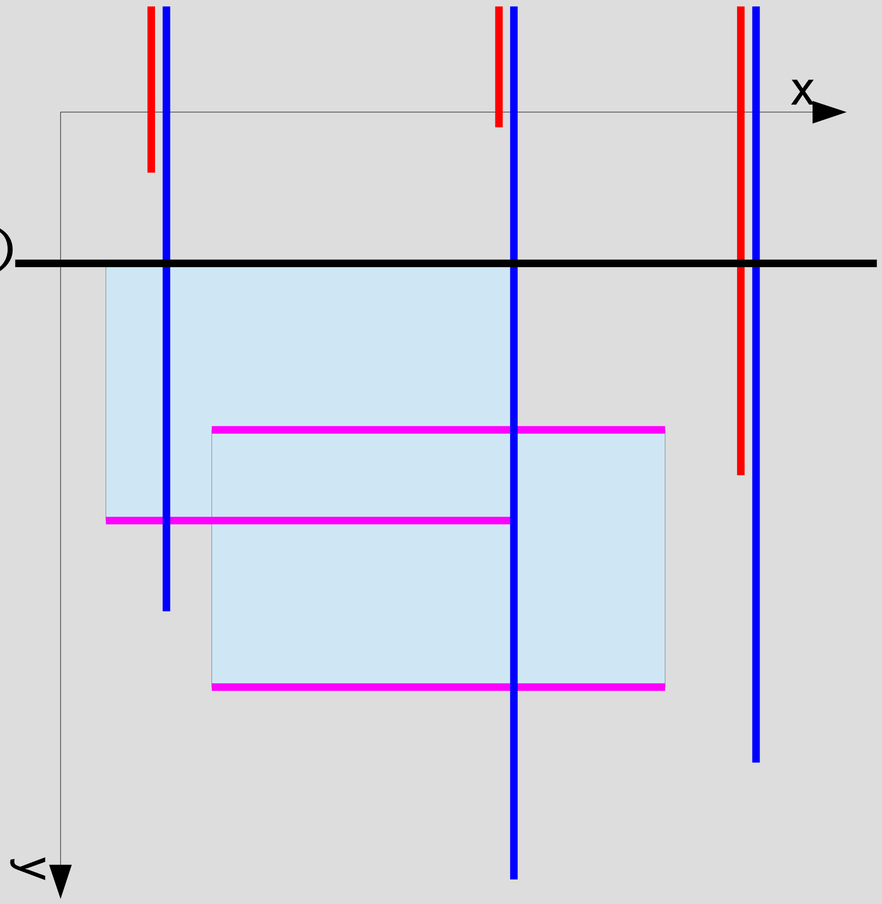
2. 長方形との交差判定(9)

- 赤線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 0



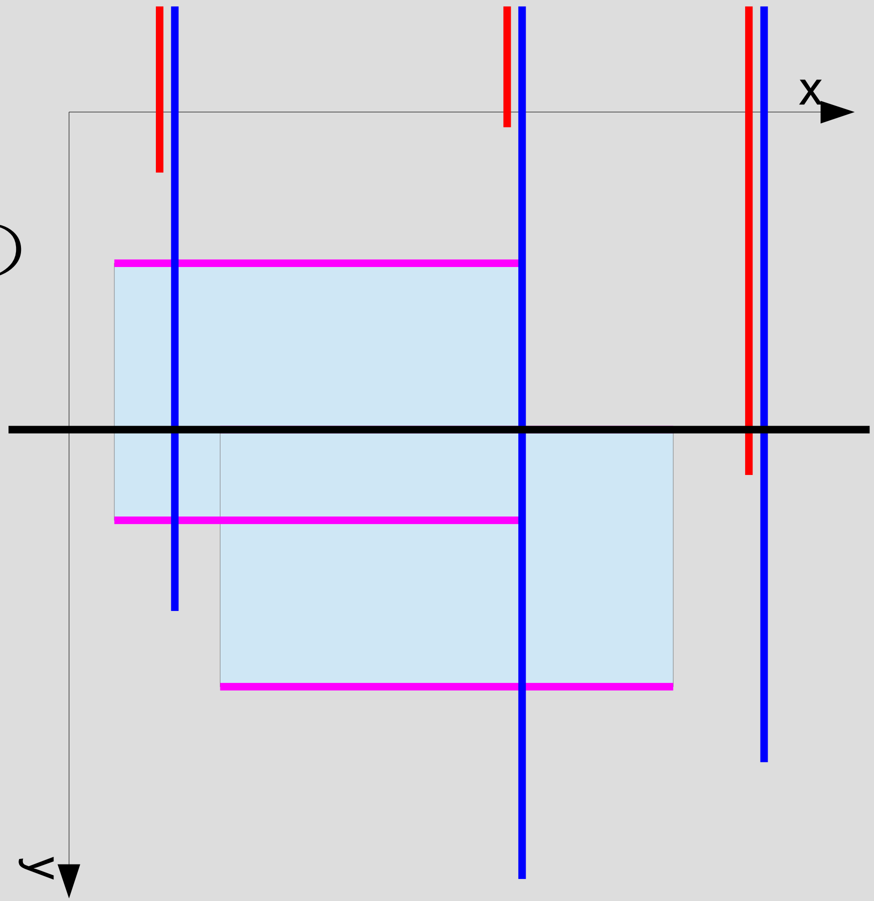
2. 長方形との交差判定(10)

- 紫線が現れた
- 紫線が占める x 座標の範囲で、出てきた紫線の数を 1 加算



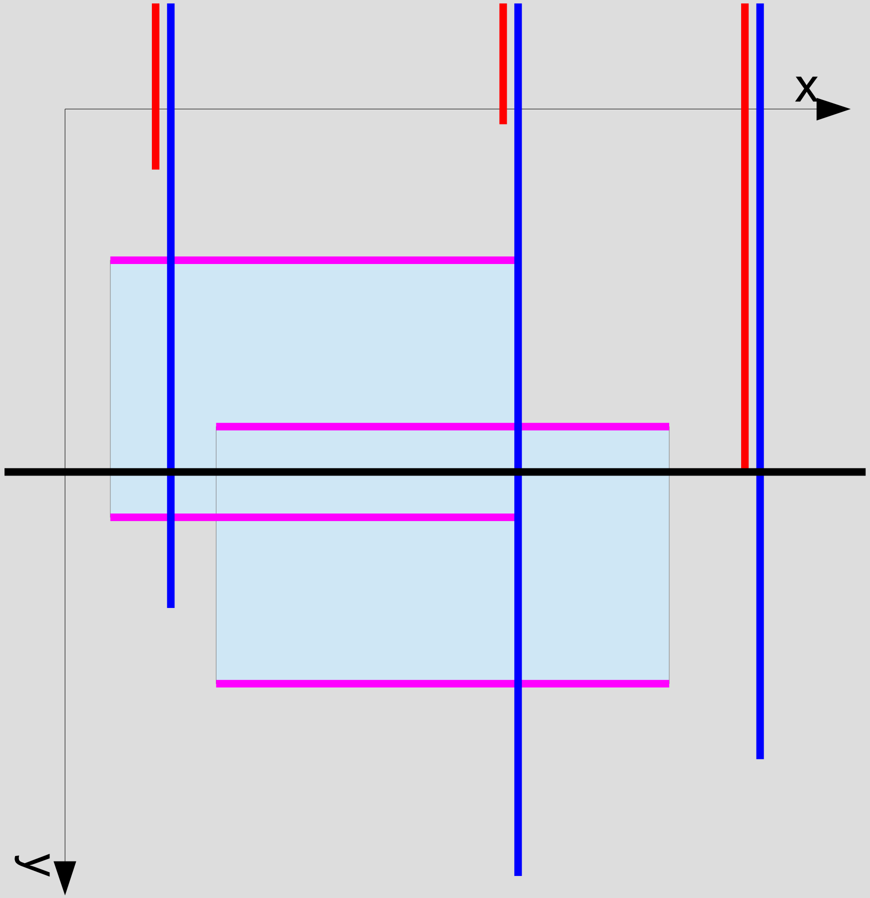
2. 長方形との交差判定(11)

- 紫線が現れた
- 紫線が占める x 座標の範囲で、出てきた紫線の数を 1 加算



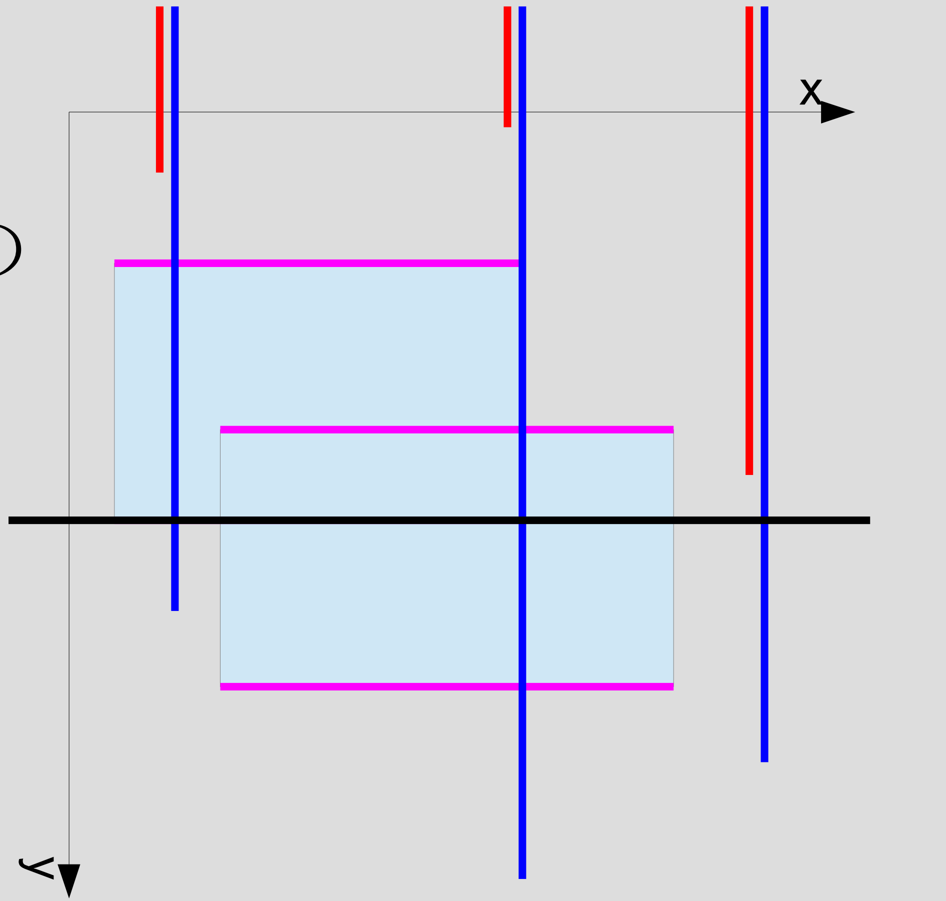
2. 長方形との交差判定(12)

- 赤線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 0



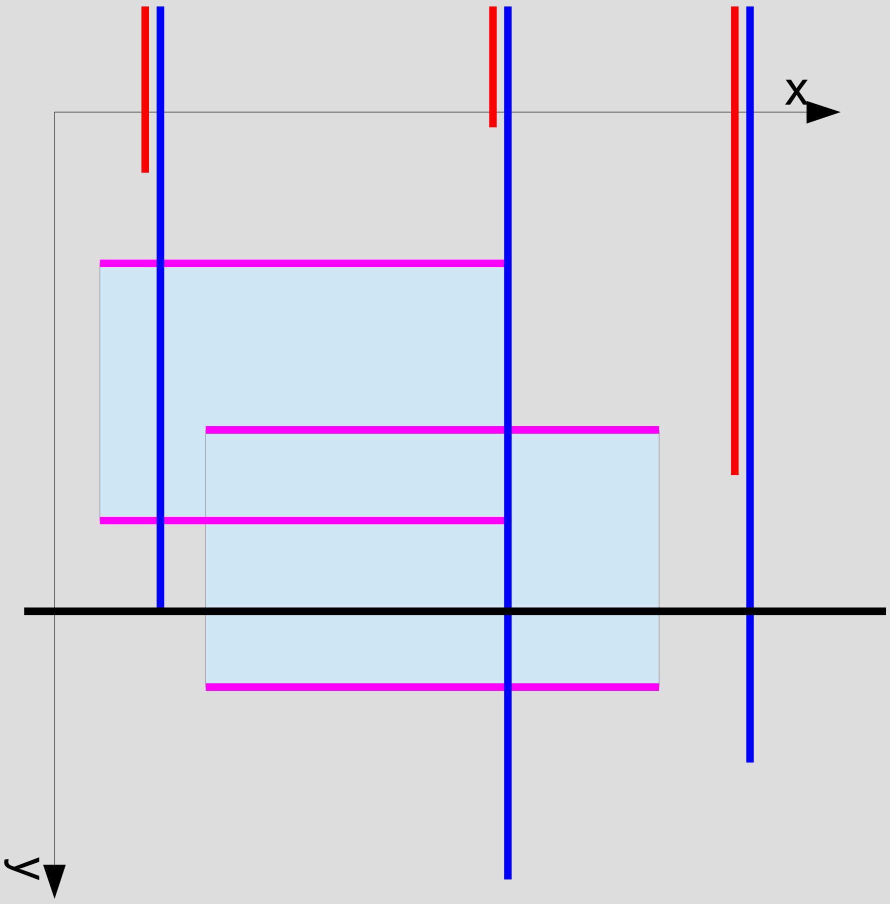
2. 長方形との交差判定(13)

- 紫線が現れた
- 紫線が占める x 座標の範囲で、出てきた紫線の数を 1 加算



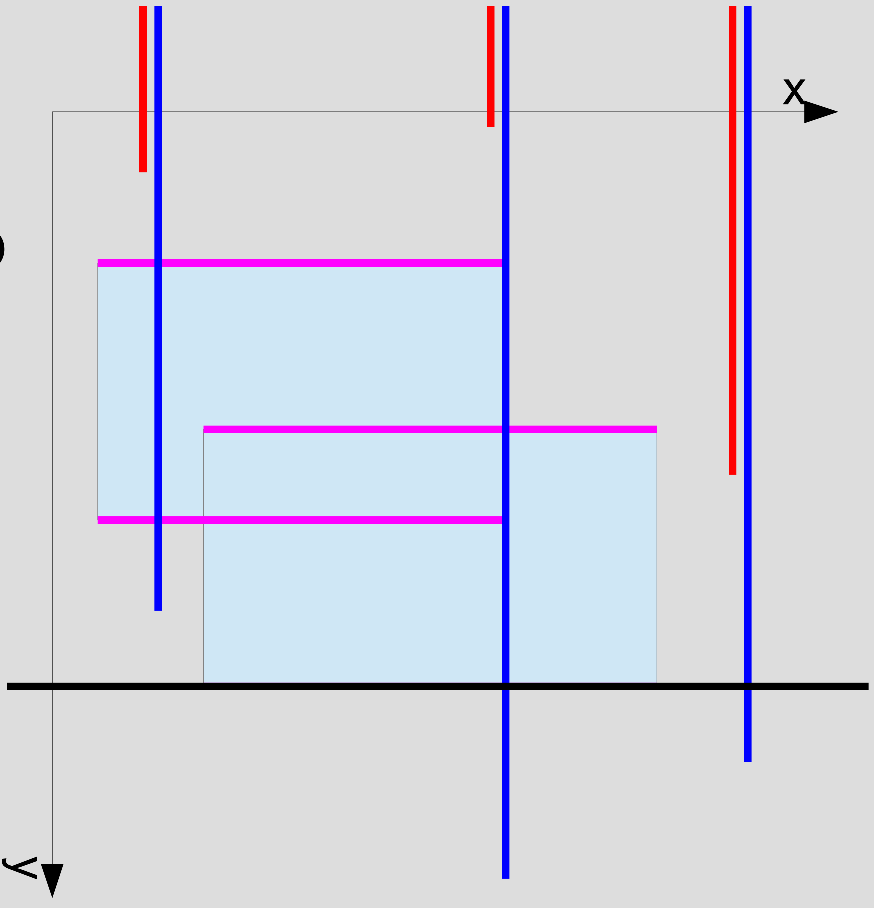
2. 長方形との交差判定(14)

- 青線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 2



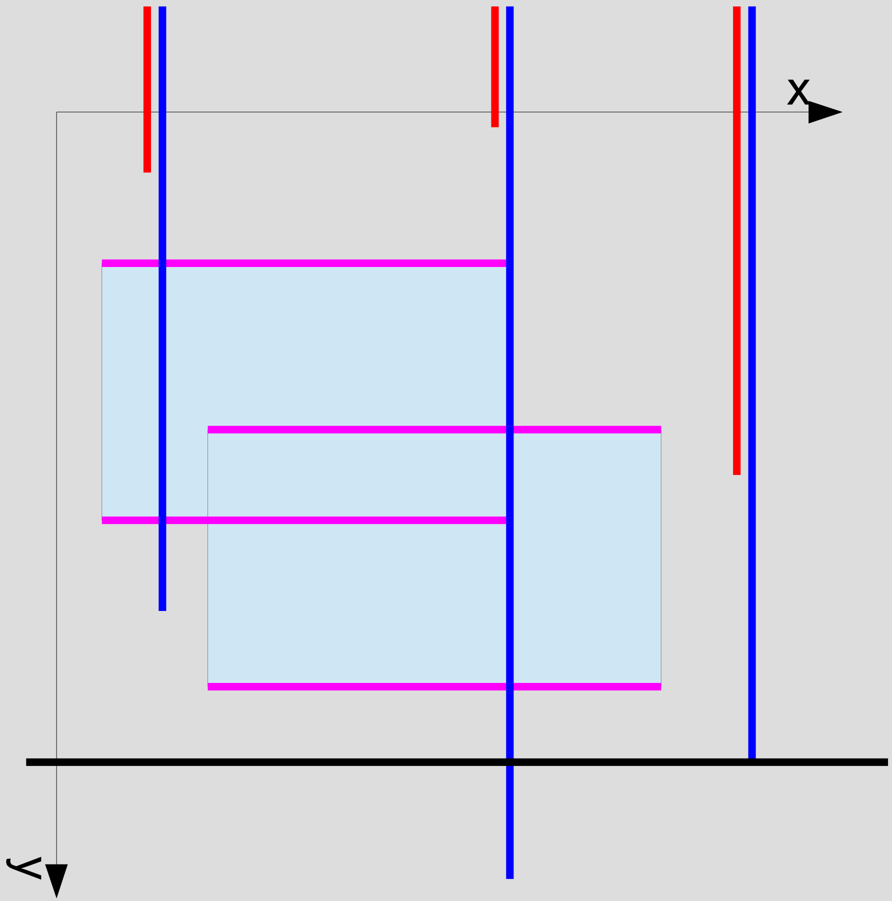
2. 長方形との交差判定(15)

- 紫線が現れた
- 紫線が占める x 座標の範囲で、出てきた紫線の数を 1 加算



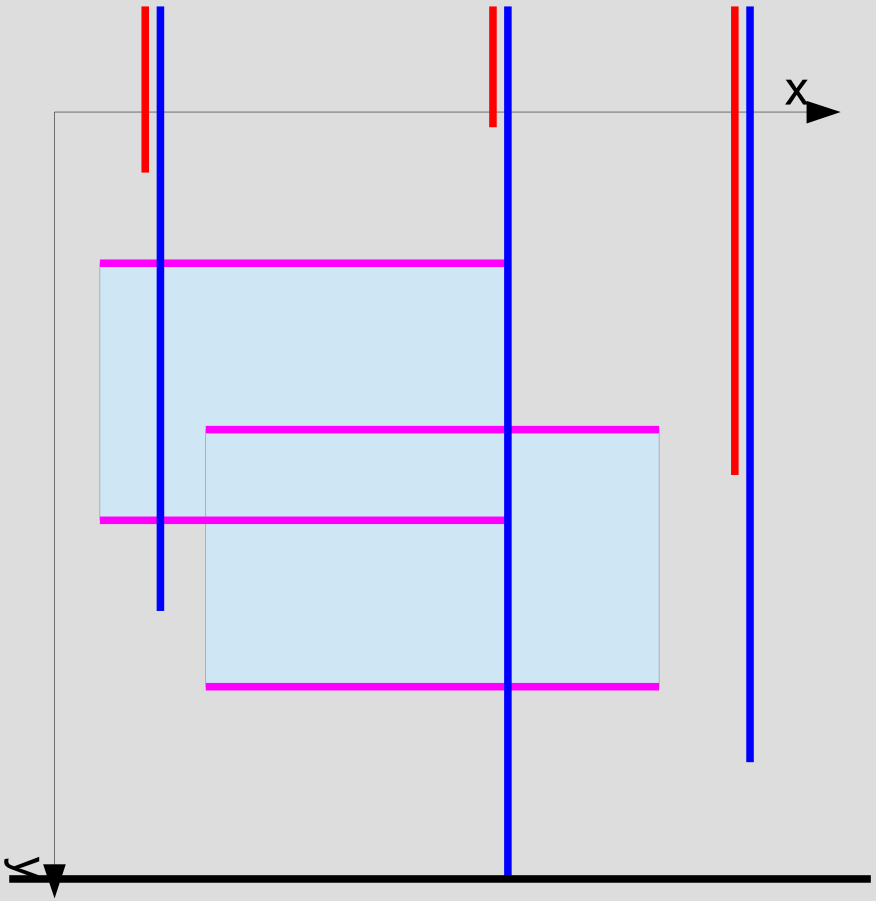
2. 長方形との交差判定(16)

- 青線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 0



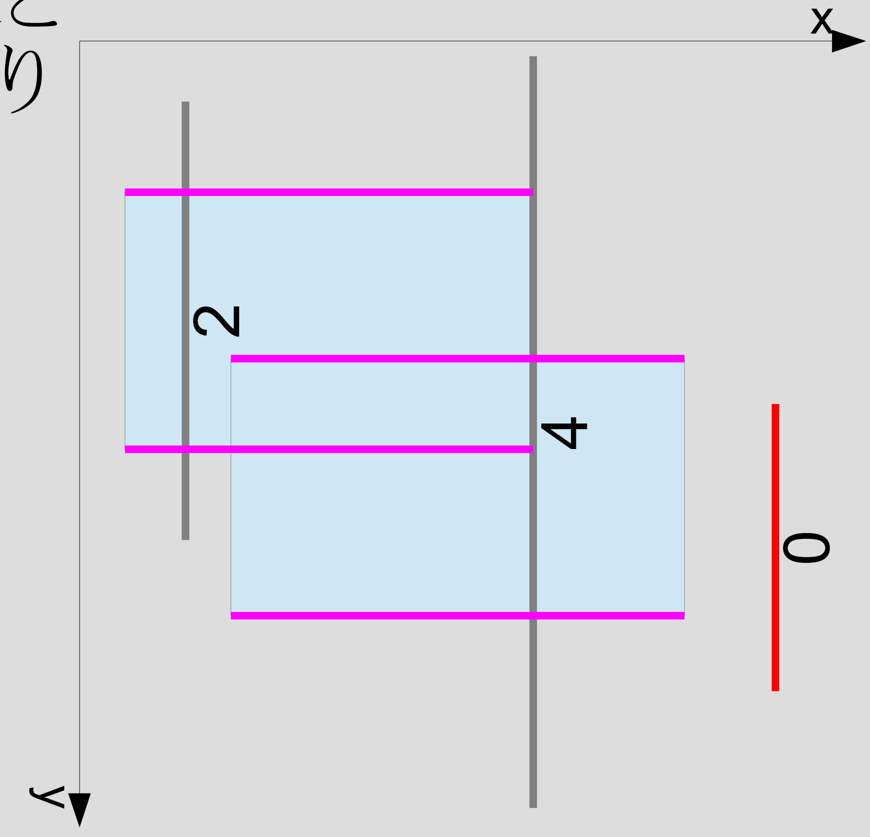
2．長方形との交差判定(17)

- 青線の下端が現れた
- この x 座標で今までに見かけた紫線の数: 4



2. 長方形との交差判定(18)

- それぞれの道路が紫線と交わる回数は右図の通り
- 0 なものだけ建設できる



2. 長方形との交差判定(19)

- 16 ページ前で「後述」とした、出てきた紫線の数を数える方法が未解決
- ここでやりたいことは、
  - x 座標の、ある範囲にすべて 1 を加算
  - ある点の値を求める

  の 2 つを高速に計算すること
- まさか、愚直に足したりするわけにはいかない！

2. 長方形との交差判定(20)

- x 座標が 1,000,000,000 もあって大変
  → 現れる x 座標をソートし、座標圧縮
- 座標圧縮の詳細については省略
  - 多分蟻本に載っています
  - 今年度の予選の 5 の解説も参照
- x 座標は N+2M 個くらいしか現れない
- 高々 600,000 個

2. 長方形との交差判定(21)

- 要素数高々 600,000 個の配列で、「区間に 1 を足す」「ある場所の値を求める」をしたい
- !!! Binary Indexed Tree(BIT) !!!
- BIT の実装は蟻本に書いてあるので省略します
- a,a+1,…,b の位置に 1 を加えるときには、 v[a]+=1, v[b+1]-=1;
- すると、位置 x の値は v[0]+…+v[x] で求まる

2. 長方形との交差判定(22)

- 以上の計算による計算量は？
  - 走査線がぶつかる座標(N+2M個)をソート
    O((N+M)log(N+M))
  - N+2M回のそれぞれで、BITにO(1)回アクセス
    O((N+M)log(N+M))
- よってO((N+M)log(N+M))

- この問題のような、「平面走査＋データ構造」はJOIでよく見る手法です
  ex.Starry Sky(09sp), Fortune Telling(12sp)

3. 道路のよい建設方法 (1)

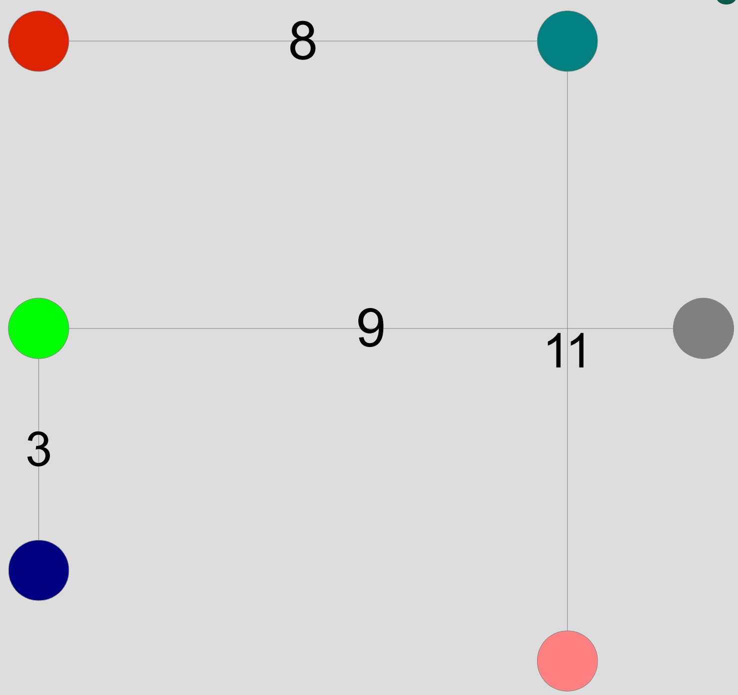
- i 個の空港を建設するときの道路コストは？

- N-i+1 個、互いに道路で結ばれた部分が必要
- 各部分に 1 つずつ空港を置く

3. 道路のよい建設方法(2)

- 最小全域木問題に似ている？
- Kruskal 法(の少し応用)
- Kruskal 法の過程で記録すると、N-i+1 個の連結成分にするための最小コストが簡単にわかる
- 辺の数はO(N) なので、Kruskal の計算量は O(NlogN) (辺のソート)

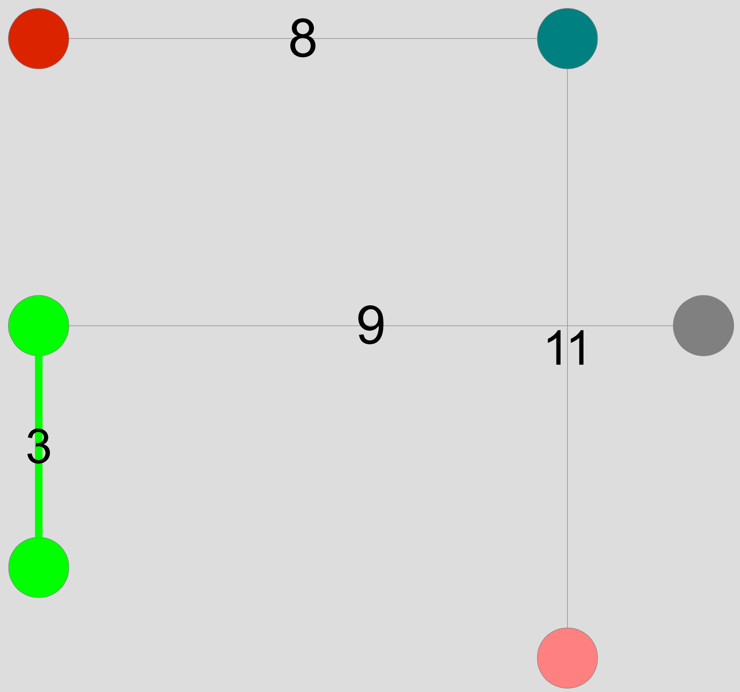
3. 道路のよい建設方法 (3)



- 短い辺から順に使う

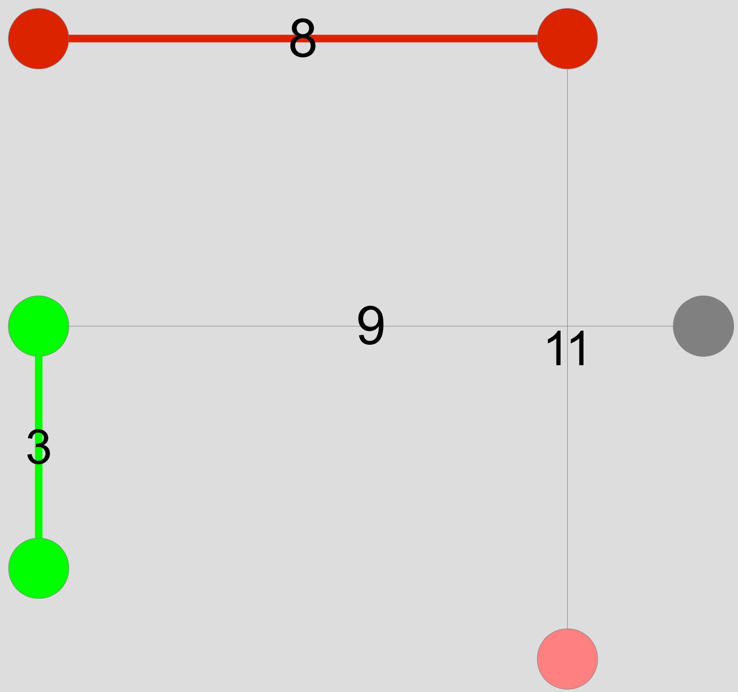
連結成分数	最小コスト
1	?
2	?
3	?
4	?
5	?
6	0

3. 道路のよい建設方法(4)



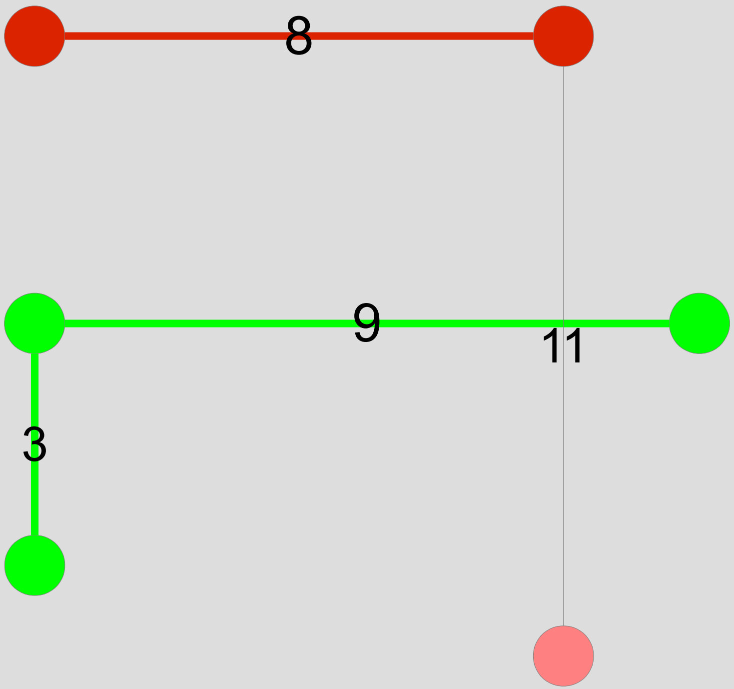
連結成分数	最小コスト
1	?
2	?
3	?
4	?
5	3
6	0

3. 道路のよい建設方法 (5)



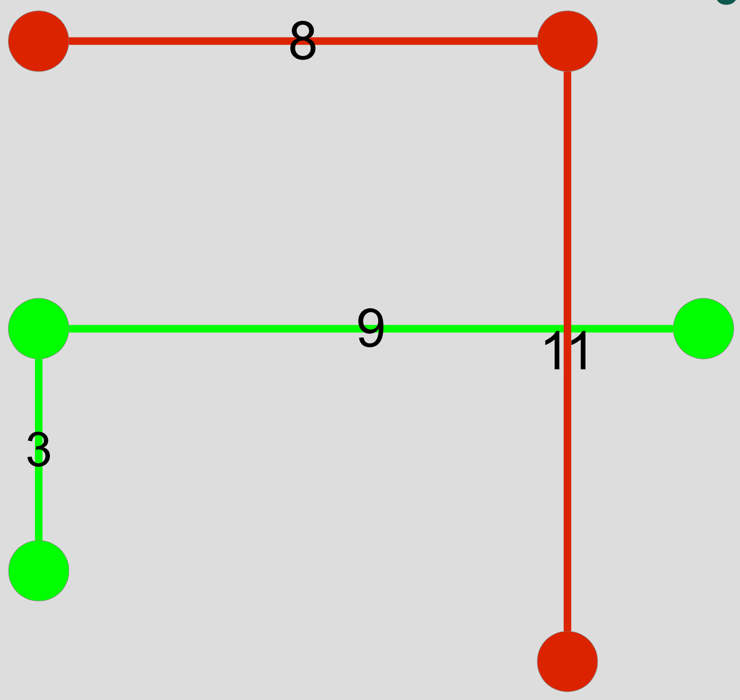
連結成分数	最小コスト
1	?
2	?
3	?
4	11
5	3
6	0

3. 道路のよい建設方法 (6)



連結成分数	最小コスト
1	?
2	?
3	20
4	11
5	3
6	0

3. 道路のよい建設方法 (7)



- これ以上連結できない

連結成分数	最小コスト
1	∞
2	31
3	20
4	11
5	3
6	0

4. 各社の建設コスト(1)

- 普通にやると O(NC) だけどもっと速くしたい
- i 個の空港を建設するときの道路コストを cost[i] とする
- C[i]=B*i+cost[i] を考えると、1<=i<=H での C[i] の最小値が答え

4. 各社の建設コスト(2)

- C[i+1]-C[i]=B-(cost[i]-cost[i+1])
- cost[i] は、i 個の連結成分を作る場合のコスト
- cost[i]-cost[i+1] は、連結成分の個数を i+1->i にするとき増やす辺のコスト
- Kruskal のアルゴリズムの動作から、辺を増やせば増やすほど新たに必要なコストも増していく

4. 各社の建設コスト(3)

- つまり、cost[i]-cost[i+1] は単調減少
- これが B 以上である限り i を大きくしてよい
- 二分探索を使うと適切な i を求められる
- 建設できる空港が少なすぎる場合に注意

- これで計算量は O(ClogN)

4. 各社の建設コスト(4)

- また例(さっきのグラフと同じ)を挙げます

i	cost[i]	cost[i]-cost[i+1]
1	∞	∞
2	31	11
3	20	9
4	11	8
5	3	3
6	0	—

4. 各社の建設コスト(5)

- B=10とすると、cost[i]-cost[i+1]が10未満になるのはi=3
- Hを無視するとi=3が最適？

i	cost[i]	cost[i]-cost[i+1]
1	∞	∞
2	31	11
3	20	9
4	11	8
5	3	3
6	0	―

4. 各社の建設コスト(6)

- 実際に C[i] を計算してみます
- やっぱり i=3 が最善

i	cost[i]	cost[i]-cost[i+1]	C[i]
1	∞	∞	∞
2	31	11	51
3	20	9	50
4	11	8	51
5	3	3	53
6	0	―	60

まとめ

- ナイーブにやるとO(NM+CN)
  - 10点(Subtask 1)
- 2.を頑張るとO((N+M)log(N+M)+CN)
  - 40点(Subtask 1,2)
- 4.を頑張るとO(NM+ClogN)
  - 40点(Subtask 1,3)
- 両方頑張るとO((N+M)log(N+M)+ClogN)
  - 100点(Subtask 1,2,3,4)

得点分布