応用アルゴリズム演習
―Topic2: 探索―

鎌田十三郎

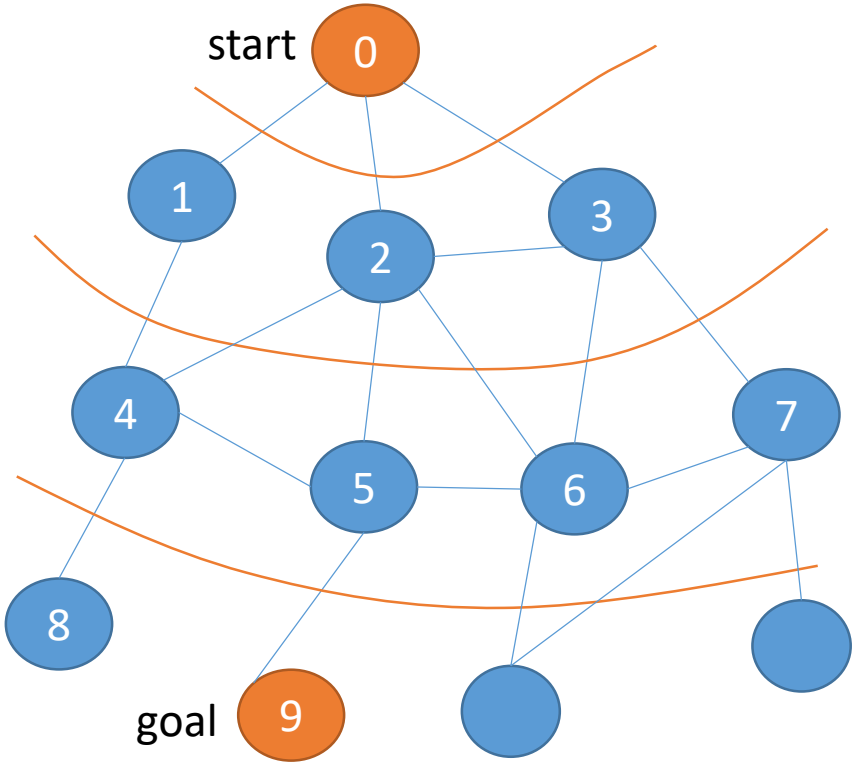
今回のテーマ

- アルゴリズム：幅優先探索

- プログラミング
  - Queue の利用：鎌田の作成した部品をあげます
  - 構造体の利用：例として座標データを扱う

今回のお題

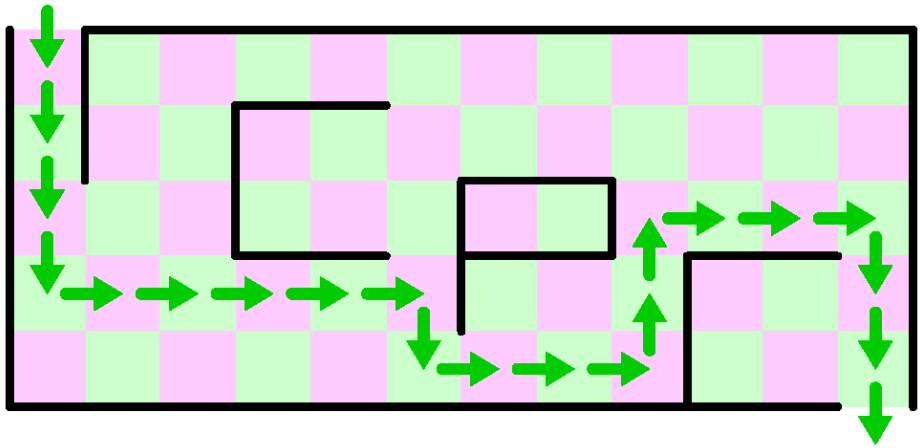
- 幅優先探索で問題を解いてみよう
  - start から goal まで、何ステップでいけるか考えよう！




```
int solve() {
    enqueue(queue, 0 );
    while(qSize(queue)>0) {
        node_t here = dequeue(queue);
        if(ゴール?) return 答え;
        未訪問の隣接node があれば、
        push(queue, node);
    }
}
```

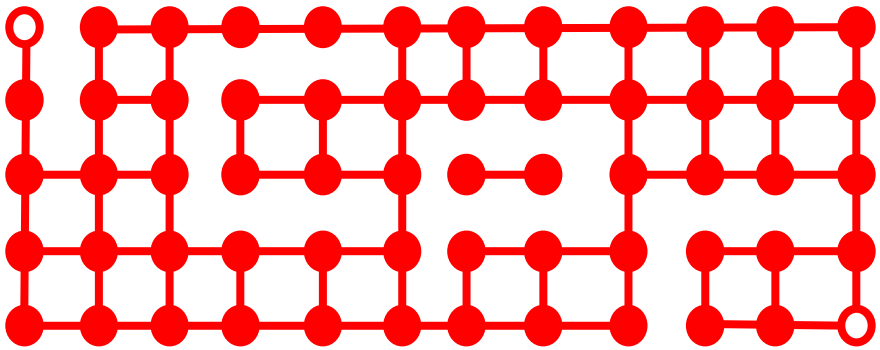
今回のお題

- 幅優先探索で問題を解いてみよう
  - 問題：ACM ICPC というプログラミングコンテストの2010年国内予選問題（参考 1, 2）



本日のお題

- 幅優先探索で問題を解いてみよう
  - 問題：ACM ICPC というプログラミングコンテストの2010年国内予選問題



```
int solve(....) {
    enqueue(queue, start);
    while(qSize(queue)>0) {
        node_t here = dequeue(queue);
        if(北にいける？
          && 未訪問？) {
            if(ゴール？) return 答え;
            訪問記録。
            enqueue(queue, north);
        }
        // 南、西、東にも同様
    }
    // 辿りつかなかった場合の処理
}
```

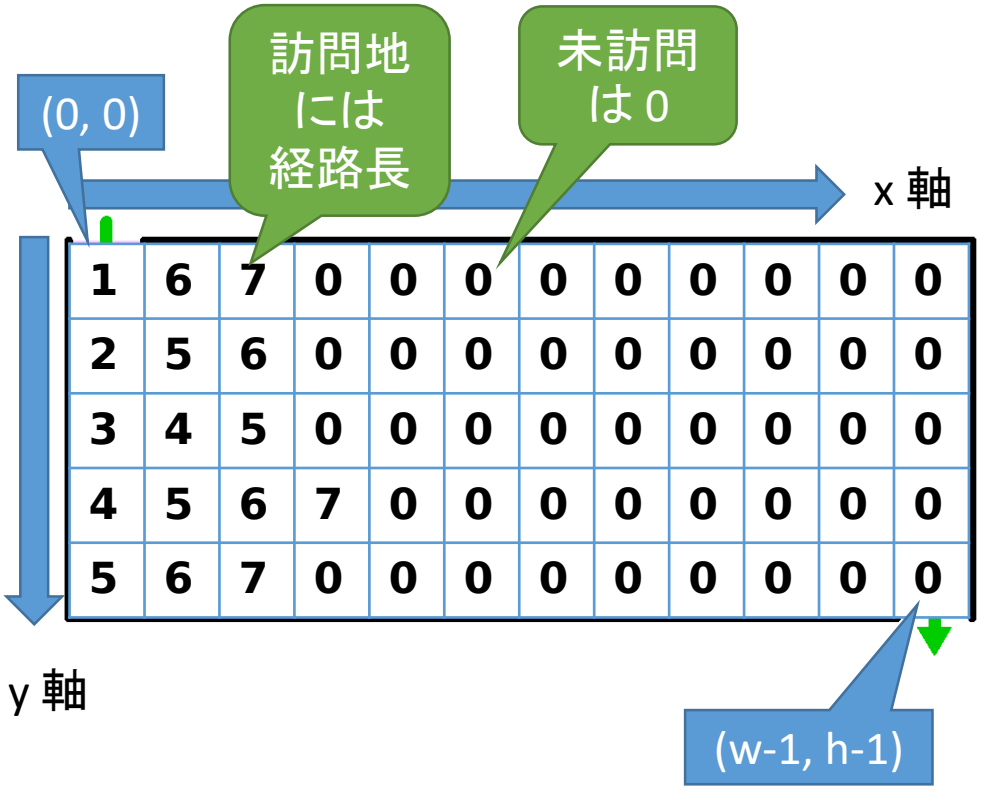
本日のお題

- なにがいるかな？
  - 各方向に進めるか？
    - 完成済み
      (解説はweb,
      program中に)
  - queue がいる
    - ライブラリあげる
  - 各地点は訪問済み？
    ステップ数の記録は？
    - 2次元配列でOK

```
int solve(....) {
   enqueue(queue, start);
   while(qSize(queue)>0) {
      node_t here = dequeue(queue);
      if(北にいける？
        && 未訪問？) {
          if(ゴール？) return 答え;
          訪問記録。
          enqueu(queue, north);
       }
      // 南、西、東にも同様
   }
   // 辿りつかなかった場合の処理
}
```

本日のお題

- なにがいるかな？
  - 各方向に進めるか？
    - 完成済み
  - queue がいる
    - ライブラリあげる
  - 各地点は訪問済み？
    ステップ数の記録は？
    - 2次元配列でOK



部品の紹介

- 座標

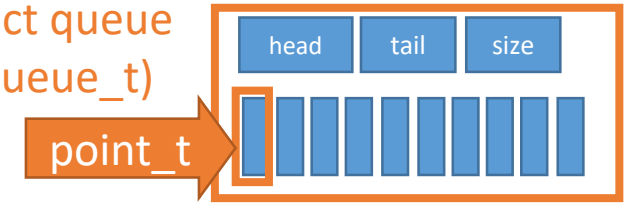
struct point
(= point_t)


```
struct point {
    int x, y;
} point_t, * point_tp;
```

- Queue
  - int qSize(queue);
  - void enqueue(queue, point);
  - point_t dequeue(queue);

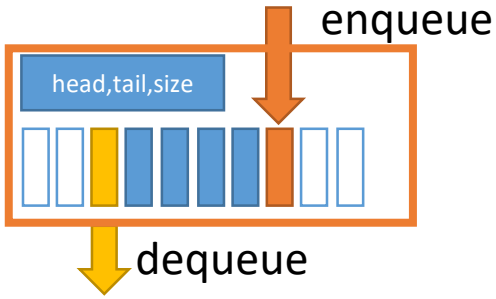
```
#define BUFSIZE 1024
struct queue {
    int head, tail, size;
    point_t buf[SIZE];
} queue_t, * queue_tp;
```

struct queue
(= queue_t)



Queue (待ち行列)

- void enqueue(q, point);



- point_t dequeue(q);

```
void enqueue(queue_tp q,
                    point_t p) {
    if(q->size >= BUFSIZE) ...;
    q->buf[q->tail++] = p;
    if(q->tail==BUFSIZE)
        q->tail = 0;
}
```

今回の実装は ring buffer を使用

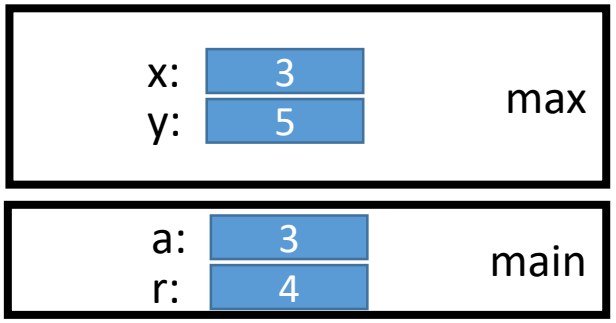
```
point_t dequeue(queue_tp q) {
    if(q->size <= 0) ....;
    point_t result =
       q->buf[q->head++];
    if(q->head==BUFSIZE) ...;
    return result;
}}
```

注：構造体を値として扱う (1/2)

- max(3, 5)

```
int main(void) {
    int a = 3;
    int r = max(a, 5);
}
```

整数値は、値をコピー

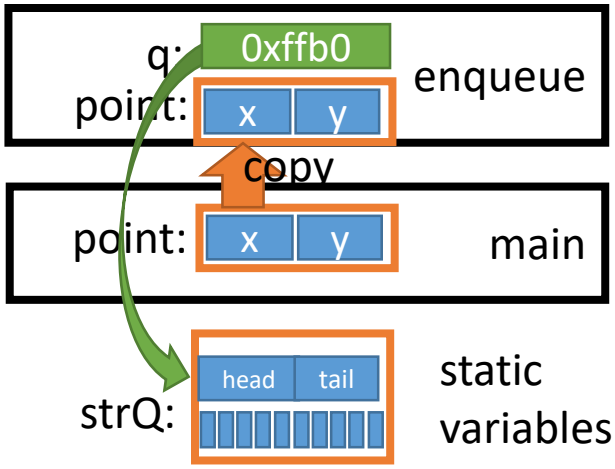


- enqueue(q, point);

```
queue_t strQ;
int main(void) {
    point_t point = {3, 5};
    enqueuer(&strQ, point);
} queue_t strQ;
```

構造体でも値はコピー

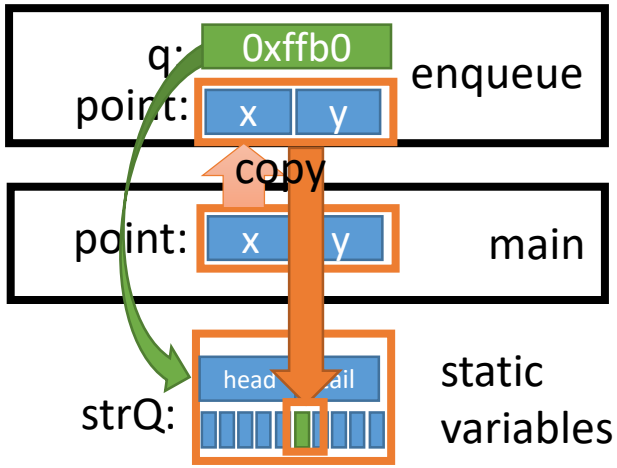
でも、Queue はポインタで管理



注：構造体を値として扱う（2/2）

- void enqueue(q, point);

```
void enqueue(queue_tp q,
                    point_t point) {
    q->buf[q->tail++] = point;
    ....
}
```



- point_t dequeue(q)

```
point_t dequeue(queue_tp q) {
    point_t result = q->buf[q->head++];
    return result;
}
int main (void) {
    point_t next = dequeue(&strQ);
}
```