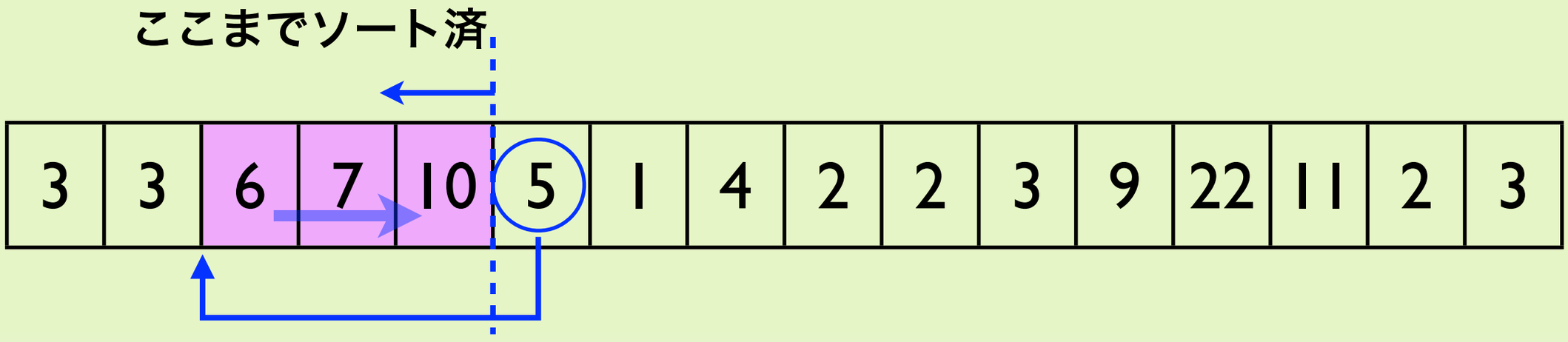
本日の課題

クイックソートを**不完全に実行する**．具体的には，大きさがm以下の配列については，ソートしないでそのまま結果を返すことにする．このようなソートをqsort_m(lst, m)としてつぎのスライドのように定義する．このように不完全にソートされた列は，挿入法（インサーションソート）でソートすると高速にソートできることが知られている．

qsort_mと挿入法を用いて**完全なソート**を行うプログラムkadai(lst)を作れ．挿入法は以下のような単純なアルゴリズムである．



5をここに挿入する．5を別の場所にコピーしたあと3の後ろスペースを作るためににピンクの部分を右に一つずらす

qsort_mのプログラム

qsort_mのプログラムは以下のとおり.

```
function qsort_m(lst, m){
    if (lst.length < m) return lst
    else {
        var key = lst[0]
        var lt = []
        var eq = []
        var gt = []
        for (var i = 0; i < lst.length; i++){
            var ele = lst[i]
            if (ele < key) lt.push(ele)
            else if (ele > key) gt.push(ele)
            else eq.push(ele)
        }
        return qsort_m(lt, m).concat(eq).concat(qsort_m(gt, m))
    }
}
```

```
print(qsort_m([5, 6, 5, 4, 3, 2, 3, 1, 8, 7, 5], 3))
```

→

```
2,1,3,3,4,5,5,5,6,8,7
```

解答例 (1)

この課題は本質的に挿入法（インサーションソート）のアルゴリズムを実現すれば良いだけである．インサーションソートのプログラムは以下のとおり．

```
function insertion_sort(lst){
    var n = lst.length;
    var i, j, k, tmp;
    for (i = 1; i < n; i++){
        j = i - 1;
        while (j >= 0 && lst[j] >= lst[i] ) j = j - 1;
        tmp = lst[i];
        for(k = i - 1; k > j; k--)
            lst[k + 1] = lst[k];
        lst[j + 1] = tmp;
    } /* for */
    return lst;
}
```

解答例 (2)

前のスライドのプログラムとsort_m(lst)を用いてkadai(lst)関数は以下のように定義される．qsort_mの第２引数は適当な値（この場合は7）にする.

```
function kadai(lst){
    return insertion_sort(qsort_m(lst, 7));
}
```

```
function test2(){
    var dat1 = [], dat2;
    var N = 100000;
    var i;
    var t1, t2, t3;
    function printit(dat){
        puts(dat[0] + "---" + dat[Math.floor(N / 2)] + "---" +dat[N - 1]);
    }
    for (i = 0; i < N; i++)
        dat1.push(Math.random());
    dat2 = dat1.slice(0);
    printit(dat1);
    t1 = new Date();
    dat1 = kadai(dat1);
    t2 = new Date();
    printit(dat1);
    dat2 = qsort_m(dat2, 1);
    t3 = new Date();
    printit(dat2);
    puts((t2 - t1) + " ms; " + (t3 - t2) + " ms");
}
```

test2()を用いてソータの性能を比較してみる.

インサーションソートを用いると高速化する

```
0.184355573848212l---0.7066564708
050482---0.707661069116636
0.0000027152999912960496---0.5002
977526760839---0.999982349189279
0.0000027152999912960496---0.5002
977526760839---0.999982349189279
458 ms; 571 ms
```