

# MA/CSSE 473

## Day 21

Permutations by  
lexicographic order  
number



# MA/CSSE 473 Day 21

- HW 9 due tomorrow;
- Don't forget the quickhull implementation problem, due Monday
- HW 10 will be due on Wednesday (Oct 15, the day before break) at noon, instead of on Tuesday
- Due to break, there will be no HW due on Tuesday, Oct 21.
- HW 11 will be due Friday, Oct 24
- Exam 2, Friday, Oct 31
- **Student Questions**
- Permutations by lexicographic order number



# Permutations and order

number	permutation	number	permutation
0	0123	12	2013
1	0132	13	2031
2	0213	14	2103
3	0231	15	2130
4	0312	16	2301
5	0321	17	2310
6	1023	18	3012
7	1032	19	3021
8	1203	20	3102
9	1230	21	3120
10	1302	22	3201
11	1320	23	3210

- Given a permutation of  $0, 1, \dots, n-1$ , can we directly find the next permutation in the lexicographic sequence?
- Given a permutation of  $0..n-1$ , can we determine its permutation sequence number?

- Given  $n$  and  $i$ , can we directly generate the  $i^{\text{th}}$  permutation of  $0, \dots, n-1$ ?



# Permutation basics

```
class Permutation:
    "Set current to the unpermuted list, and all directions pointing left"
    def __init__(self, n):
        self.current = range(1, n + 1)
        self.n = n
        self.more = True # This is not the last permutation.

    def swap(self, i, j):
        self.current[i], self.current[j] = self.current[j], self.current[i]

    def reverse(self, i, j):
        while j > i:
            self.swap(i, j)
            i += 1
            j -= 1
```



# Main permutation method

```
def next(self):  
    "return current permutation and calculate next one"  
    if not self.more:  
        return False  
    returnValue = list(self.current)  
    i = self.n-2  
    while self.current[i]>self.current[i+1]:  
        i -= 1  
    j = self.n - 1  
    while self.current[i] > self.current[j]:  
        j -= 1  
    self.swap(i,j)  
    self.reverse(i+1, self.n-1)  
    if i == -1:  
        self.more = False  
  
    return "".join([str(v) for v in returnValue])
```



# Find a permutation's sequence #

```
def factorialTable(n):  
    "return a table of the factorials of n-1, n-2, ..., 0"  
    table = [1]  
    for i in range(1,n):  
        table = [i*table[0]] + table  
    return table  
  
def permNumber(p):  
    """assumes that p is a permutation of 0..n-1.  
    returns k such that p is the kth lexicographic  
    permutation of those numbers."""  
    p = list(p) # make a copy  
    n = len(p)  
    factList = factorialTable(n)  
    sum = 0  
    for i in range(n):  
        next = p[i]  
        sum += p[i] * factList[i]  
        for j in range(i + 1, n):  
            if p[j] > p[i]:  
                p[j] -= 1  
    return sum
```

