2/7/17 ①

Odometer

0	0	0	0
0	0	0	1

⋮

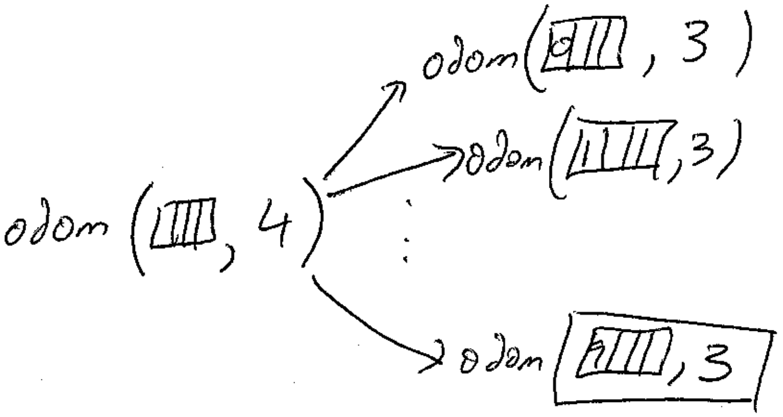
0	0	0	9
0	0	1	0

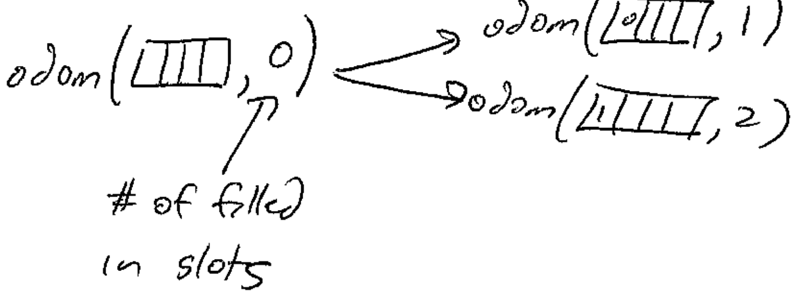
⋮

0	0	9	9
0	1	0	0
0	9	9	9
1	0	0	0

⋮

9	9	9	9





2/7/17 ②

Subsets

{0, 1, 2}

	~~3~~ 2	1	0
{ } →	0	0	0
{0}	0	0	1
{1}	0	1	0
{2}	1	0	0
{0,1} →	0	1	1
{0,2}	1	0	1
{1,2}	1	1	0
{1,2,0}	1	1	1

Permutations

0, 1, 2
0, 2, 1
1, 0, 2
1, 2, 0
2, 0, 1
2, 1, 0

Odometer but you can't reuse a digit!

0,1,2,3,4,5,6,7

array	3	1	6	0				
used	1	1	0	1	0	0	1	0
	0	1	2	3	4	5	6	7

k=3

```
for (i=0; i<n; i++) {
    if (!used[i]) {
        perm[k] = i;
        used[i] = 1;
}       permutation (perm, used, k+1, n); → used[i]=0;
```

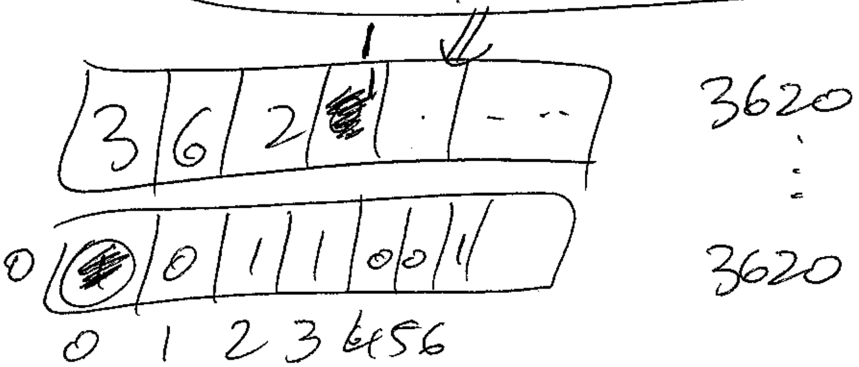
```
Void permutation(int *perm, int* used, int k, int len) {
        if (k == len) {
            //Process permutation
            return;
        }
        int i;
        for (i = 0; i < len; i++) {
            if (!used[i]) {
                perm[k] = i;
                used[i] = 1;
                permutation(perm, used, k+1, len);
                used[i] = 0; //***
            }
        }
}
```

Derangements

A derangement is a permutation where no integer is in its original place.

0, 1, 2, 3

2, 0, 3, 1 → is valid



0 3 2 4 1

$+3 \quad +1 \quad +2 \quad +3 \quad = \boxed{9}$

0 3 1 2

$3 + 2 + 1 = 6$

1 3 0 2

$2 + 3 + 2 = 7$ ✓

Derangement

A permutation where no item (number) is in its original slot.

0, 1, 2, 3

1, 0, 3, 2
1, 2, 3, 0
1, 3, 0, 2
2, 0, 3, 1
2, 3, 0, 1
2 3, 1, 0

Upwards
skip=2
length=5

$\boxed{\text{bgjrw}}$

SORTING

① Bubble Sort
② Insertion Sort
③ Selection Sort

$O(n^2)$ Sorts

④ Merge Sort (today)
⑤ Quick Sort (tuesday)

$O(n \lg n)$
On average

Good exercise.
Write bubble sort recursively

Bubble Sort

8, 6, 12, 3, 7, 1, 4

6 8 12 3 7 1 4

6 8 12 3

6 8 3 12 7 1 4

6 8 3 7 1 4 12 → now, just sort this!

One iteration of bubble sort

6 3 7 1 4 8 12 → after 2nd iteration

3 6 1 4 7 8 12

3 1 4 6 7 8 12

1 3 4 6 7 8 12

1 3 4 6 7 8 12 DONE

2/9/17 ②

Bubble Sort Run Time (n elements)

1st iteration = n steps

2nd iterat = n−1 steps

3rd iter = n−2 step

⋮

last iter = 1 step

$$\text{Run-time} = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2) \checkmark$$

```
Recursive BubbleSort(Array, n) {
        if (n == 1) return;
  →     //runOnePass (for loop n times)
  ⇒     BubbleSort(Array, n−1);
}
```

Let T(n) = run time of bubble sort

$$T(n) = n + T(n-1) \quad , T(1) = 1$$

$$= n + (n-1) + T(n-2)$$

$$= n + (n-1) + (n-2) + T(n-3)$$

$$= \sum_{i=n-k+1}^{n} i + T(n-k)$$

Let $k = n-1$

$$= \sum_{i=2}^{n} i + T(1)$$

$$= \sum_{i=2}^{n} i + 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} = O(n^2)$$

A common recurrence relation: 2/9/17 ③

$$T(n) = f(n) + T(n-1) \Longleftarrow$$

$$= \underline{f(n)} + \underline{f(n-1)} + \underline{f(n-2)} + T(n-3)$$

$$= \sum_{i=1}^{n} f(i) + \boxed{T(0)}$$

Insertion Sort

```
for each item i=1 to n-1 :
    insert item i into its correct
    location in the already sorted array
    [0, i-1].
    while
```

Analysis

Best Case: already sorted $O(n)$

Worst Case: Reverse $O(n^2)$

Avg Case: $\frac{1}{2}$ of worst case $O(n^2)$

Selection Sort

```
Run n-1 times:
{for i=n-1 ; i>0; i--) {
    //Find index of the max item from
    // index 0 to i.
    Swap maxindex with i.
}
```

$$n + (n-1) + (n-2) + \vdots + 1$$

$$= \sum_{i=1} i$$

$$= \frac{n(n+1)}{2}$$

$$= O(n^2)$$

2/9/17 ④

Merse Sort

Merge Algorithm

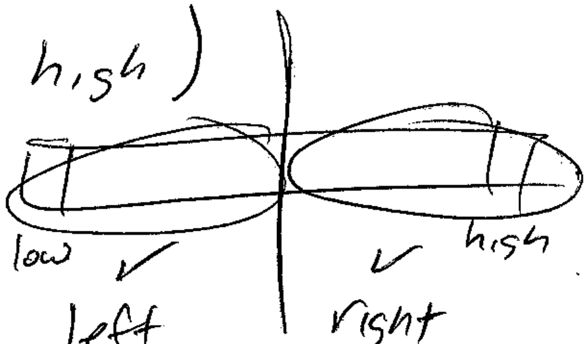
list 1: 2, 4, 5, 9, 15, 18, 27, 33   n elements

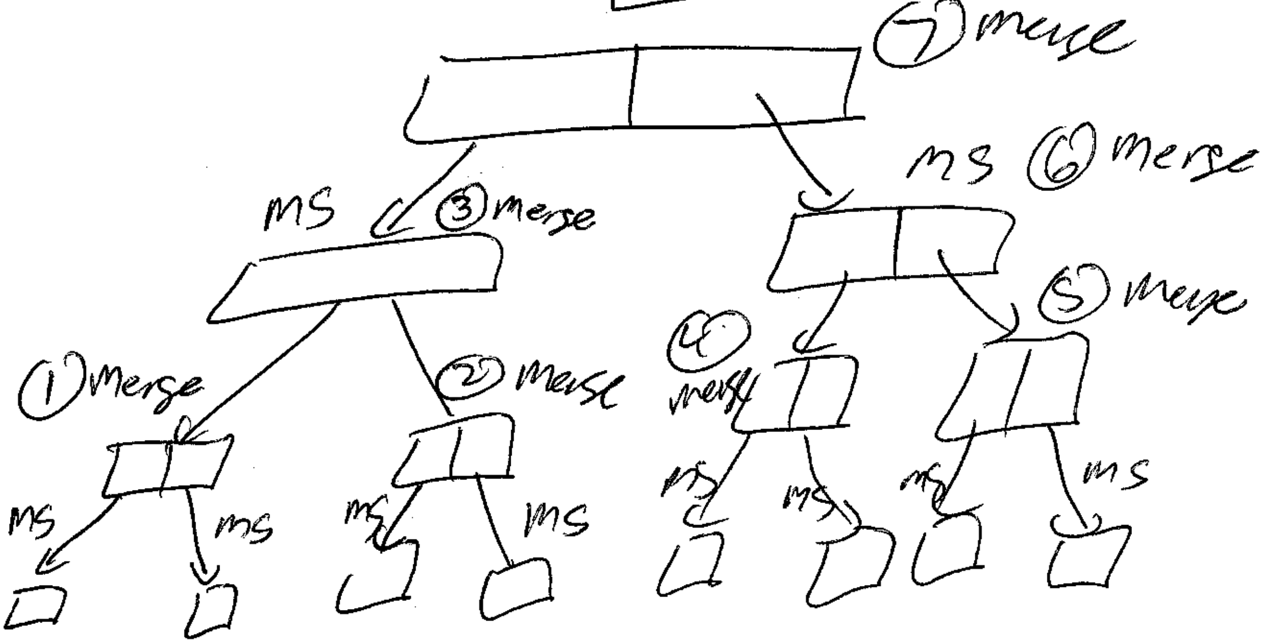
list 2: 3, 6, 10, 11, 12, 22   m elements

list 3: 2, 3, 4, 5, 6, 9, 10, 11, 12, 15, 18, 22, 27, 33

$O(n+m)$

```
MergeSort(array, low, high)
    int mid = (low+high)/2;
    MergeSort(array, low, mid);
    MergeSort(array, mid+1, high);
    ~~return~~ Merge(array, low, mid, mid+1, high)
```


low
high
left
right


⑦ merse
ms
⑥ merse
ms
③ merse
⑤ merse
④ merse
① merse
② merse
ms
ms
ms
ms
ms
ms
ms
ms