

MA/CSSE 473

Day 20

Combinatorial
Object Generation



MA/CSSE 473 Day 20

- HW 8 due now; HW9 Friday
- Don't forget the quickhull implementation problem, due next 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
- **Student Questions**
- Combinatorial Object Generation



Pass around attendance sheet.

Some Homework Algorithms

- Nuts and Bolts
- Ferrying Soldiers
- Celebrity Identification



Lexicographic Permutation Generation

- Generate the permutations in "natural" order.
- Let's do it (recursively)



Solution on next Slide

Lexicographic Permutation Code

```
def permuteRecursive(prefix, remaining):  
    if remaining == []:  
        return [prefix]  
    result = []  
    for n in remaining:  
        copy = [e for e in remaining]  
        copy.remove(n)  
        result += permuteRecursive(prefix + [n], copy)  
    return result  
  
def permute(n):  
    return permuteRecursive([], range(1, n+1))  
  
print permute(3)
```



The book describes a "permute in place" algorithm.

All Subsets of a Set

- Sample Application:
 - Solving the knapsack problem
 - In the brute force approach, we try all subsets
- If A is a set, the set of all subsets is called the **power set** of A , and often denoted 2^A
- If A is finite, then $|2^A| = 2^{|A|}$
- So we know how many subsets we need to generate.



Generating Subsets of $\{a_1, \dots, a_n\}$

- Decrease by one:
- Generate S_{n-1} , the collection of the 2^{n-1} subsets of $\{a_1, \dots, a_{n-1}\}$
- Then $S_n = S_{n-1} \cup \{S_{n-1} \cup \{a_n\} : s \in S_{n-1}\}$
- Another approach:
 - Each subset of $\{a_1, \dots, a_n\}$ corresponds to a bit string of length n , where the i^{th} bit is 1 iff a_i is in the subset



Another approach:

- Each subset of $\{a_1, \dots, a_n\}$ corresponds to a bit string of length n , where the i^{th} bit is 1 if and only if a_i is in the subset

```
def allSubsets(s):  
    n = len(s)  
    subsets=[]  
    for i in range(2**n):  
        subset = []  
        current = i  
        for j in range (n):  
            if current % 2 == 1:  
                subset += [s[j]]  
            current /= 2  
        subsets += [subset]  
    return subsets
```

Output:

```
[[], [1],  
[2], [1, 2],  
[3], [1, 3],  
[2, 3],  
[1, 2, 3]]
```



Gray Codes

- Named for Frank Gray
- An ordering of the 2^n n-bit binary codes such that any two consecutive codes differ in only one bit
- Example:
000, 001, 011, 010, 110, 111, 101, 100
- Note also that only one bit changes between the last code and the first code.
- A Gray code can be represented by its **transition sequence**: which bit changes each time
In above example: 0, 1, 0, 2, 0, 1, 0
- In terms of subsets, the transition sequence tells which element to add or remove from one subset to get the next subset



Generating a Gray Code

- Binary Reflected Gray Code
- $T_1 = 0$
- $T_{n+1} = T_n, n, T_n^{\text{reversed}}$
- Show by induction that $T_n^{\text{reversed}} = T_n$
- Thus $T_{n+1} = T_n, n, T_n$

