DESIGN AND ANALYSIS OF ALGORITHMS

Surabhi Narayan
Department of Computer Science & Engineering

DESIGN AND ANALYSIS OF ALGORITHMS

DECREASE AND CONQUER

Surabhi Narayan
Department of Computer Science & Engineering

Combinatorial Objects

- Permutations
- Combinations
- Subsets of a given set

Generating Permutations

- Underlying set elements are to be permuted
- Decrease and conquer approach
- Satisfies the minimal change requirement
- Example: Johnson- Trotter algorithm

Generating Permutations

ALGORITHM *JohnsonTrotter(n)*

//Implements Johnson-Trotter algorithm for generating permutations
//Input: A positive integer n
//Output: A list of all permutations of $\{1, \ldots, n\}$
initialize the first permutation with $\overleftarrow{1}\,\overleftarrow{2} \ldots \overleftarrow{n}$
while the last permutation has a mobile element **do**
    find its largest mobile element k
    swap k with the adjacent element k's arrow points to
    reverse the direction of all the elements that are larger than k
    add the new permutation to the list

Decrease and Conquer

$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{3}\,\overleftarrow{\mathbf{4}}$	$\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{\mathbf{4}}\,\overleftarrow{3}$	$\overleftarrow{1}\,\overleftarrow{\mathbf{4}}\,\overleftarrow{2}\,\overleftarrow{3}$	$\overleftarrow{4}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{\mathbf{3}}$
$\overrightarrow{\mathbf{4}}\,\overleftarrow{1}\,\overleftarrow{3}\,\overleftarrow{2}$	$\overleftarrow{1}\,\overrightarrow{\mathbf{4}}\,\overleftarrow{3}\,\overleftarrow{2}$	$\overleftarrow{1}\,\overleftarrow{3}\,\overrightarrow{\mathbf{4}}\,\overleftarrow{2}$	$\overleftarrow{1}\,\overleftarrow{\mathbf{3}}\,\overleftarrow{2}\,\overrightarrow{4}$
$\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{2}\,\overleftarrow{\mathbf{4}}$	$\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{\mathbf{4}}\,\overleftarrow{2}$	$\overleftarrow{3}\,\overleftarrow{\mathbf{4}}\,\overleftarrow{1}\,\overleftarrow{2}$	$\overleftarrow{4}\,\overleftarrow{3}\,\overleftarrow{1}\,\overleftarrow{\mathbf{2}}$
$\overrightarrow{\mathbf{4}}\,\overrightarrow{3}\,\overleftarrow{2}\,\overleftarrow{1}$	$\overrightarrow{3}\,\overrightarrow{\mathbf{4}}\,\overleftarrow{2}\,\overleftarrow{1}$	$\overrightarrow{3}\,\overleftarrow{2}\,\overrightarrow{\mathbf{4}}\,\overleftarrow{1}$	$\overrightarrow{\mathbf{3}}\,\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{4}$
$\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{1}\,\overleftarrow{\mathbf{4}}$	$\overleftarrow{2}\,\overrightarrow{3}\,\overleftarrow{\mathbf{4}}\,\overleftarrow{1}$	$\overleftarrow{2}\,\overleftarrow{\mathbf{4}}\,\overrightarrow{3}\,\overleftarrow{1}$	$\overleftarrow{4}\,\overleftarrow{2}\,\overrightarrow{\mathbf{3}}\,\overleftarrow{1}$
$\overrightarrow{\mathbf{4}}\,\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3}$	$\overleftarrow{2}\,\overrightarrow{\mathbf{4}}\,\overleftarrow{1}\,\overrightarrow{3}$	$\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{\mathbf{4}}\,\overrightarrow{3}$	$\overleftarrow{2}\,\overleftarrow{1}\,\overrightarrow{3}\,\overrightarrow{4}$

Permutations in Lexicographic order

ALGORITHM *LexicographicPermute*(n)

//Generates permutations in lexicographic order
//Input: A positive integer n
//Output: A list of all permutations of $\{1, \ldots, n\}$ in lexicographic order
initialize the first permutation with $12 \ldots n$
while last permutation has two consecutive elements in increasing order **do**
  let i be its largest index such that $a_i < a_{i+1}$ //$a_{i+1} > a_{i+2} > \cdots > a_n$
  find the largest index j such that $a_i < a_j$ //$j \geq i + 1$ since $a_i < a_{i+1}$
  swap a_i with a_j //$a_{i+1}a_{i+2} \ldots a_n$ will remain in decreasing order
  reverse the order of the elements from a_{i+1} to a_n inclusive
  add the new permutation to the list

Generating Subsets:

Knapsack problem needed to find the most valuable subset of items that fits a knapsack of a given capacity.

Powerset: set of all subsets of a set. Set A={1, 2, ..., n} has $\mathbf{2^n}$ subsets.

Generate all subsets of the set A={1, 2, ..., n}.

Any **decrease-by-one** idea?
\# of subsets of { } = 2^0 = 1, which is **{ }** itself
Suppose, we know how to generate all subsets of {1,2,...,n-1}
Now, how can we generate all subsets of {1,2,...,n} ?

Generating Subsets:

All subsets of {1,2,...,n-1}: 2^{n-1} such subsets

All subsets of {1,2,...,n}:
2^{n-1} subsets of {1,2,...,n-1} and
another 2^{n-1} subsets of {1,2,...,n-1} having **'n'** with them.

That adds up to all 2^n subsets of {1,2,...,n}

0	$\varnothing$							
1	$\varnothing$	$\{a_1\}$						
2	$\varnothing$	$\{a_1\}$	$\{a_2\}$	$\{a_1, a_2\}$				
3	$\varnothing$	$\{a_1\}$	$\{a_2\}$	$\{a_1, a_2\}$	$\{a_3\}$	$\{a_1, a_3\}$	$\{a_2, a_3\}$	$\{a_1, a_2, a_3\}$

Alternate way of Generating Subsets:

Knowing the binary nature of either having **n**th element or not, any idea involving binary numbers itself?

One-to-one correspondence between all 2^n bit strings $b_1b_2...b_n$ and 2^n subsets of $\{a_1, a_2, \ldots, a_n\}$.

Each bit string $b_1b_2...b_n$ could correspond to a subset.
In a bit string $b_1b_2...b_n$, depending on whether b_i is 1 or 0, a_i is in the subset or not in the subset.

000	001	010	011	100	101	110	111
$\varnothing$	$\{a_3\}$	$\{a_2\}$	$\{a_2, a_3\}$	$\{a_1\}$	$\{a_1, a_3\}$	$\{a_1, a_2\}$	$\{a_1, a_2, a_3\}$

Generating Subsets in **Squashed order:**

Squashed order: any subset involving a_j can be listed only after all the subsets involving $a_1, a_2, \ldots, a_{j-1}$

Both of the previous methods does generate subsets in squashed order.

000	001	010	011	100	101	110	111
$\varnothing$	$\{a_3\}$	$\{a_2\}$	$\{a_2, a_3\}$	$\{a_1\}$	$\{a_1, a_3\}$	$\{a_1, a_2\}$	$\{a_1, a_2, a_3\}$

Generating Subsets in **Squashed order:**

Squashed order: any subset involving a_j can be listed only after all the subsets involving a_1, a_2, …, a_{j-1}

Can we do it with minimal change in bit-string (actually, just one-bit change to get the next bit string)? This would mean, to get a new subset, just change one item (remove one item or add one item).

Binary reflected gray code:

000 001 011 010 110 111 101 100

THANK YOU

Surabhi Narayan
Department of Computer Science &Engineering

surabhinarayan@pes.edu