

Chapter 5: Decrease and Conquer

A journey of a thousand miles begins with a single step. (Confucius)

Introduction	2
Decrease by a Constant	3
Decrease by a Constant Factor	4
Insertion Sort	5
Insertion Sort Algorithm	5
Illustration.	6
Comments on Insertion Sort.	7
Depth-First Search and Breadth-First Search	8
Graph Traversal	8
Graph Marking Algorithm	9
Depth-First Search Algorithm.	10
Breadth-First Search Algorithm	11
DFS Illustration	12
Comments on Graph Traversal Algorithms	13
Miscellaneous Problems	14
Gray Code Problem	14
The Fake Coin Problem.	15
Multiplication a la Russe	16
The Josephus Problem	17
The Selection Problem	18

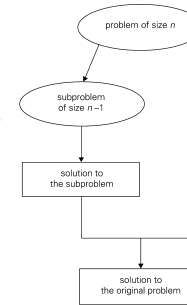
Introduction

Decrease-and-conquer is an approach to solving a problem by:

- Change an instance into one smaller instance of the problem.
- Solve the smaller instance.
- Convert the solution of the smaller instance into a solution for the larger instance.

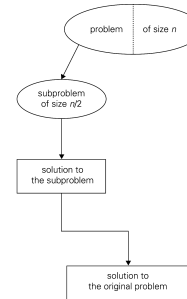
Decrease by a Constant

Decrease-by-a-constant decreases the instance size by 1 (or some other constant), e.g., $2^{10} = 2 * 2^9$



Decrease by a Constant Factor

Decrease-by-a-constant-factor decreases the instance size by half (or some other fraction), e.g., $2^{10} = 2^5 * 2^5$.



Insertion Sort

5

Insertion Sort Algorithm

```
algorithm InsertionSort( $A[0..n-1]$ )
// Sorts a given array by insertion sort
// Input: An array  $A$  of orderable elements
// Output: Array  $A[0..n-1]$  in ascending order
for  $i \leftarrow 1$  to  $n-1$  do
     $v \leftarrow A[i]$ 
     $j \leftarrow i-1$ 
    while  $j \geq 0$  and  $A[j] > v$  do
         $A[j+1] \leftarrow A[j]$ 
         $j \leftarrow j-1$ 
     $A[j+1] \leftarrow v$ 
```

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 5

Illustration

```
89 | 45 68 90 29 34 17
45 89 | 68 90 29 34 17
45 68 89 | 90 29 34 17
45 68 89 90 | 29 34 17
29 45 68 89 90 | 34 17
29 34 45 68 89 90 | 17
17 29 34 45 68 89 90
```

FIGURE 5.4 Example of sorting with insertion sort. A vertical bar separates the sorted part of the array from the remaining elements; the element being inserted is in bold.

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 6

3

Comments on Insertion Sort

- Insertion sort ensures $A[0] \leq A[1] \leq \dots \leq A[i-1]$.
- Insertion sort looks for correct position for $A[i]$.
- Insertion sort shifts values at and above correct position.
- Worst Case: The number of comparisons is $\sum_{i=1}^{n-1} i = n(n-1)/2 \in O(n^2)$.
- Best Case: $n-1 \in \Omega(n)$ comparisons if array is already sorted.
- Average Case $\approx n^2/4$ comparisons.

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 7

Depth-First Search and Breadth-First Search

8

Graph Traversal

- Graph traversal algorithms process all the vertices of a graph in a systematic fashion.
- They are useful for many graph problems such as checking connectivity, checking acyclicity, connected components, finding articulation points, and topological sorting.
- First all the vertices are marked as unvisited.
- Then an unvisited vertex is selected, marked as visited, and all unvisited vertices reachable from that vertex are marked as visited.
- Repeat above step until all vertices are visited.

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 8

4

Graph Marking Algorithm

```
algorithm MarkGraph( $G$ )
// Marks vertices and calls traversal algorithm
// Input: Graph  $G = (V, E)$ 
// Output: Vertices marked in traversal order
mark each vertex in  $V$  with 0 // means unvisited
count  $\leftarrow 0$  // count is global
for each vertex  $v$  in  $v$  do
  if  $v$  is marked with 0
    call traversal algorithm with  $v$ 
```

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 9

Depth-First Search Algorithm

```
algorithm DFS( $v$ )
// Recursively visits unvisited vertices from  $v$ 
// Input: Vertex  $v$ 
// Output: Unvisited vertices from  $v$  are marked
count  $\leftarrow$  count + 1
mark  $v$  with count
for each vertex  $u$  adjacent from  $v$  do
  if  $u$  is marked with 0
    DFS( $u$ )
```

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 10

Breadth-First Search Algorithm

```
algorithm BFS( $v$ )
// Visits unvisited vertices from  $v$ 
// Input: Vertex  $v$ 
// Output: Unvisited vertices from  $v$  are marked
count  $\leftarrow$  count + 1; mark  $v$  with count
initialize a queue with  $v$ 
while the queue is not empty do
   $u \leftarrow$  remove vertex from the queue
  for each vertex  $w$  adjacent from  $u$  do
    if  $w$  is marked with 0
      count  $\leftarrow$  count + 1
      mark  $w$  with count
      add  $w$  to the queue
```

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 11

DFS Illustration

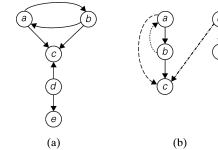


FIGURE 5.8 (a) Digraph. (b) DFS forest of the digraph for the DFS traversal started at a .

Tree Edges: ab, bc, de

Back Edges: ba

Forward Edges: ac

Cross Edges: dc

CS 3343 Analysis of Algorithms

Chapter 5: Slide – 12

Comments on Graph Traversal Algorithms

- Each vertex is marked with 0 once and some positive integer once.
- Each edge is processed once for directed graphs, twice for undirected graphs.
 - In DFS, edges from a vertex are processed after vertex is marked.
 - In BFS, edges from a vertex are processed after vertex is dequeued. Each vertex is enqueued once (right after it is marked).
- All Cases: Each vertex and edge is processed once or twice, which implies $\Theta(V + E)$.

CS 3343 Analysis of Algorithms Chapter 5: Slide – 13

Miscellaneous Problems

14

Gray Code Problem

- A Gray Code generates all possible bit strings of a given length by changing only one bit each time.
- One Gray Code for 3 bits is: 000, 001, 011, 010, 110, 111, 101, 100. This is useful for generating all subsets of length 3.
- An outline of the algorithm for length n is:
 - Generate Gray code for length $n - 1$.
 - Change n th bit and output bit string.
 - Generate Gray code for length $n - 1$.

CS 3343 Analysis of Algorithms Chapter 5: Slide – 14

The Fake Coin Problem

- The fake coin problem is to detect a single fake coin from a set of coins using a balance scale.
- Supposing the fake coin is lighter, we can repeatedly compare one half of a set to the other half, eliminating the heavier half.
- This decreases the number of coins in half each iteration, so only $\approx \log_2 n$ weighings are needed.
- Consider variations of this problem, e.g., not known whether coin is lighter or heavier, or 2 fake coins.

CS 3343 Analysis of Algorithms Chapter 5: Slide – 15

Multiplication a la Russe

To multiply two pos. integers $n \cdot m$ by this method:

$$n \cdot m = \begin{cases} m & \text{if } n = 1 \\ (n/2) \cdot 2m & \text{if } n \text{ is even} \\ (\lfloor n/2 \rfloor \cdot 2m + m) & \text{if } n \text{ is odd} \end{cases}$$

n	m		n	m	
50	65		50	65	
25	130		25	130	130
12	260	(+130)	12	260	
6	520		6	520	
3	1,040		3	1,040	1,040
1	2,080	(+1040)	1	2,080	2,080
	2,080	+(130 + 1040) = 3,250			3,250

(a) (b)

FIGURE 5.14 Computing $50 \cdot 65$ by multiplication à la russe
 CS 3343 Analysis of Algorithms Chapter 5: Slide – 16

The Josephus Problem

- In the Josephus Problem, every other number is eliminated until one is left, for example:

1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9
1		3		5	7	9		
		3	5	7	9			
		3		7	9			
		3		7	9			
- Let $J(n)$ = the last number for $1, \dots, n$.

$$J(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2J(n/2) - 1 & \text{if } n \text{ is even} \\ 2J(\lfloor n/2 \rfloor) + 1 & \text{if } n \text{ is odd} \end{cases}$$
- Computing $J(n)$ as above is $\Theta(\log_2 n)$.

CS 3343 Analysis of Algorithms Chapter 5: Slide – 17

The Selection Problem

- The Selection Problem is finding the k th smallest element out of n elements.
- One solution is to sort the elements, and then return $A[k - 1]$. This is $\Theta(n \log n)$.
- Better is to modify quicksort to only sort the partition with $A[k - 1]$. This is $\Theta(n)$ on average.
- Suppose partitioning always splits in half, then the recurrence is: $C(n) = C(n/2) + \Theta(n)$, which is $\Theta(n)$ by the Master Theorem.
- Even if $A[k - 1]$ is always in a "big" partition, say $9/10$, then $C(n) = C(9n/10) + \Theta(n)$ is also $\Theta(n)$.