Ch.3 in The Algorithm Design Manual

A) Contiguous vs. Linked Data Structures

- Data structures can be classified as either **contiguous** or **linked** depending on whether they are based on arrays of pointers

- **Contiguously-allocated structures**
  - Composed of single slabs of memory and include arrays, matrices, heaps, and hash tables

- **Linked Data Structure**
  - Composed of distince chunks of memory bound together by **pointers** and include lists, trees, and graph adjacency lists

Note:
Work through Ch.3 + then pick up CTCI

1) Arrays

- **Definition**

  Structures of fixed size data records such that each element can be efficiently located by its index or address

- **Advantages**

  a) Constant time access given the index

  b) Space EFFICIENCY
  - Arrays consist purely of data, so space is not wasted with links or formatting info

  - End of record info isn't needed since arrays are built from fixed-size records

  c) Memory Locality
  - Elements in the array are physically located next to each other which makes arrays ideal for iterating

Note:
Write the Readme for g-notes github

- **Disadvantages**

  a) Cannot adjust their size in the middle of a program's execution

1.1) Dynamic Arrays

- This allows us to efficiently enlarge arrays as we need them

- Example:
  - Suppose we have an array of size 1 and we double its size from m to 2m each time we run out of space.
    - So to double would require creating a new contiguous array of size 2m
    - Then copying the contents of the old array to the lower half of the new one and returning the space used by the old array to the storage allocation system

- It will take $\log_2 n$ doublings until the array will have n elements

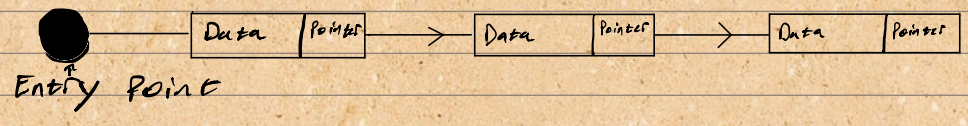
- Worst case runtime for managing dynamic arrays is $O(N)$

- What we lose using dynamic arrays is the guarantee that each array access takes constant time in the worst case
  - All queries are fast except the few that trigger a redoubling

2) Pointers and Linked Structures
- Pointers hold pieces of linked structures together

- Pointers represents the address of a location in memory

- Linked list



a) Each node contains one or more data fields that hold the actual data we need to store

b) Each node also contains a pointer field to atleast one other node

- Doubly linked list have 2 pointers.

c) We also need a pointer ~~to the~~ starting point of the list

d) 3 basic methods:
- searching
- Insertion
- Deletion