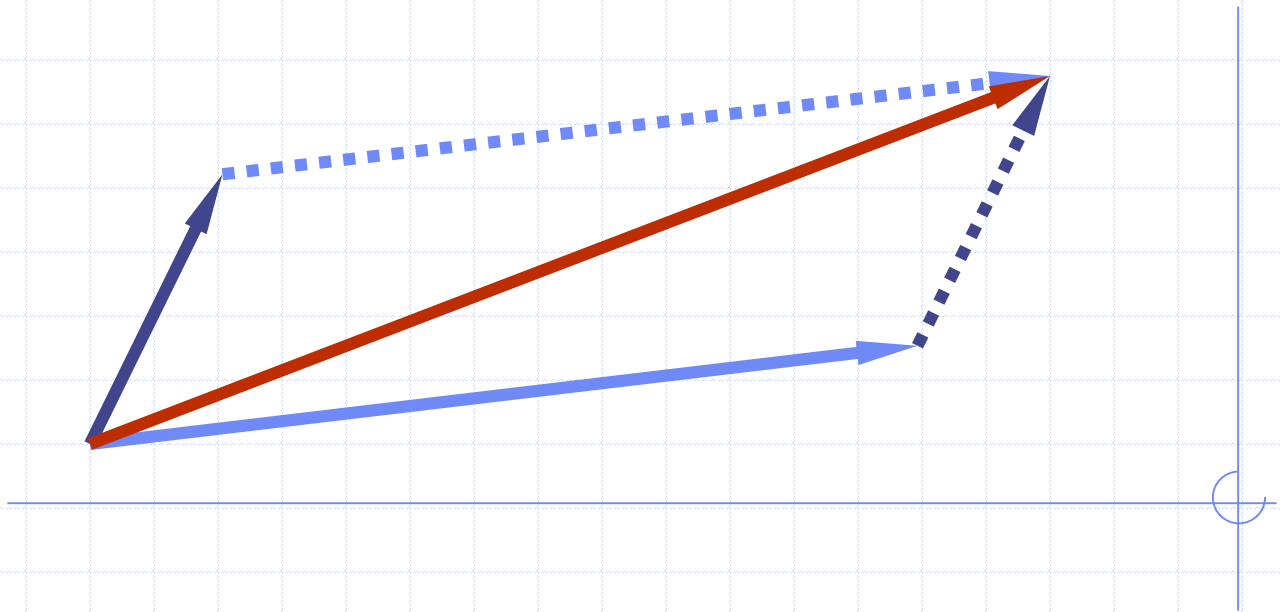


Vectors



Outline and Reading

- ◆ The Vector ADT (§2.2.1)
- ◆ Array-based implementation (§2.2.1)

The Vector ADT

- ◆ The **Vector** ADT extends the notion of array by storing a sequence of arbitrary objects
- ◆ An element can be accessed, inserted or removed by specifying its rank (number of elements preceding it)
- ◆ An exception is thrown if an incorrect rank is specified (e.g., a negative rank)
- ◆ Main vector operations:
 - object **elemAtRank**(integer r): returns the element at rank r without removing it
 - object **replaceAtRank**(integer r, object o): replace the element at rank r with o and return the old element
 - **insertAtRank**(integer r, object o): insert a new element o to have rank r
 - object **removeAtRank**(integer r): removes and returns the element at rank r
- ◆ Additional operations **size()** and **isEmpty()**

Applications of Vectors

◆ Direct applications

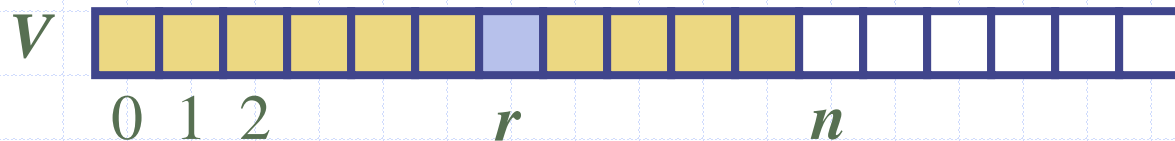
- Sorted collection of objects (elementary database)

◆ Indirect applications

- Auxiliary data structure for algorithms
- Component of other data structures

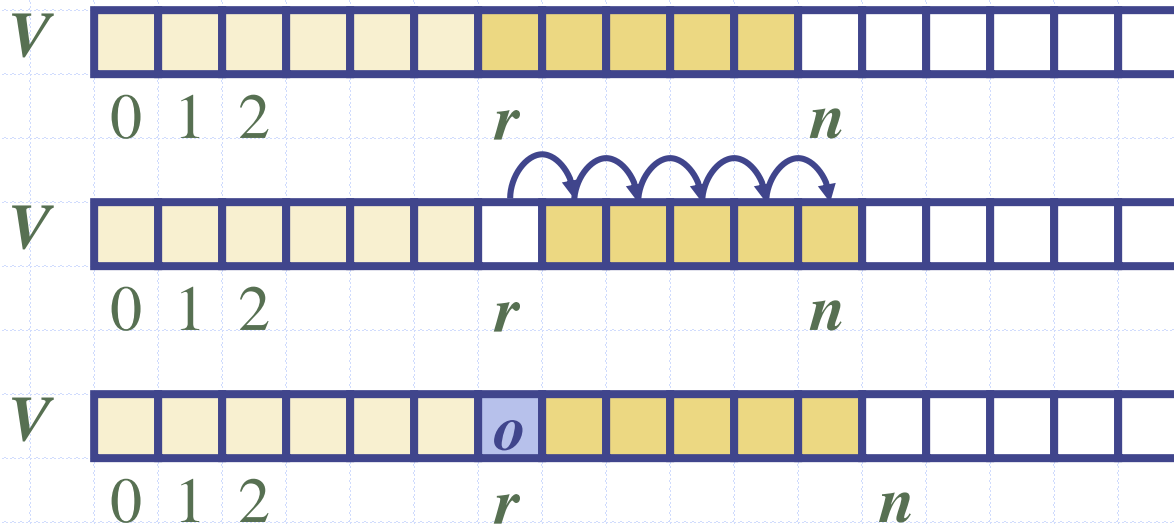
Array-based Vector

- ◆ Use an array V of size N
- ◆ A variable n keeps track of the size of the vector (number of elements stored)
- ◆ Operation *elemAtRank*(r) is implemented in $O(1)$ time by returning $V[r]$



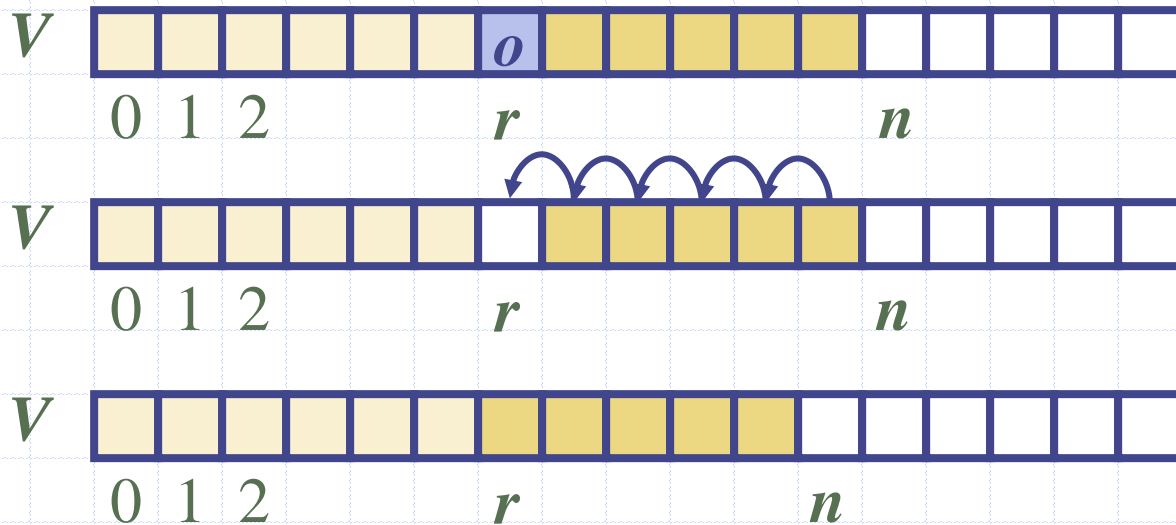
Insertion

- ◆ In operation *insertAtRank*(r, o), we need to make room for the new element by shifting forward the $n - r$ elements $V[r], \dots, V[n - 1]$
- ◆ In the worst case ($r = 0$), this takes $O(n)$ time



Deletion

- ◆ In operation *removeAtRank*(r), we need to fill the hole left by the removed element by shifting backward the $n - r - 1$ elements $V[r + 1], \dots, V[n - 1]$
- ◆ In the worst case ($r = 0$), this takes $O(n)$ time



Performance

- ◆ In the array based implementation of a Vector
 - The space used by the data structure is $O(n)$
 - *size*, *isEmpty*, *elemAtRank* and *replaceAtRank* run in $O(1)$ time
 - *insertAtRank* and *removeAtRank* run in $O(n)$ time
- ◆ If we use the array in a circular fashion, *insertAtRank*(0) and *removeAtRank*(0) run in $O(1)$ time
- ◆ In an *insertAtRank* operation, when the array is full, instead of throwing an exception, we can replace the array with a larger one