Data Structures and Algorithms

Lecture 13

Aniket Basu Roy

BITS Pilani Goa Campus

2026-02-14 Sat

Agenda

Data Structures

Data Structures

- What and Why

Data Structures

- What and Why
- a way to store data/information and
- a way to rerieve
- a way to delete
- a way to create relationship among data, e.g., precedence, successor, etc.

Data Structures

- Linear: Arrays, Stacks, Queues, Deques, Linked Lists
- Non-Linear: Heaps, Binary Search Trees, Graphs
- Hashing: The magic of $O(1)$ lookup

Example

A computer game that stores a deck of playing cards.

A few queries we want to answer:

Example

A computer game that stores a deck of playing cards.

A few queries we want to answer:

1. We want to add a card into the deck.
2. Which card is at the top of the deck?
3. Is the deck empty?
4. Given a card are there any higher rank cards of the same suit in the deck. E.g., Given 9♣, is there a card $C \in$ {10♣, J♣, Q♣, K♣} in the deck?

Stacks

Push(a)

Pop()

TopElement()

IsEmpty()

Stacks

```
STACK-EMPTY(S)
if S.top == 0
    return TRUE
else return FALSE

PUSH(S, x)
if S.top == S.size
    error "overflow"
else S.top = S.top + 1
    S[S.top] = x

POP(S)
if STACK-EMPTY(S)
    error "underflow"
else S.top = S.top - 1
    return S[S.top + 1]
```

Queues

Enqueue(a)

Dequeue()

IsEmpty()

Queues

```
ENQUEUE(Q, x)
Q[Q.tail] = x
if Q.tail == Q.size
        Q.tail = 1
else Q.tail = Q.tail + 1

DEQUEUE(Q)
x = Q[Q.head]
if Q.head == Q.size
        Q.head = 1
else Q.head = Q.head + 1
return x
```

Linked Lists

Node

- prev
- key
- next

Operations

- Search
- Prepend
- Insert
- Delete

Linked Lists

LIST-SEARCH(L, k)

```
x = L.head
while x ≠ NIL and x.key ≠ k
    x = x.next
return x
```

Linked Lists

```
LIST-PREPEND(L, x)
  x.next = L.head
  x.prev = NIL
  if L.head ≠ NIL
      L.head.prev = x
  L.head = x
```

Linked Lists

```
LIST-INSERT(x, y)
  x.next = y.next
  x.prev = y
  if y.next ≠ NIL
      y.next.prev = x
  y.next = x
```

Linked Lists

LIST-DELETE(L, x)

```
if x.prev ≠ NIL
    x.prev.next = x.next
else L.head = x.next
if x.next ≠ NIL
    x.next.prev = x.prev
```

Stacks - A deeper dive

Given $\{1, 2, \ldots, n\}$, how many permutations can we generate using a stack?