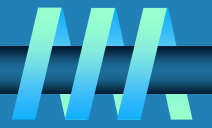
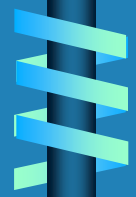
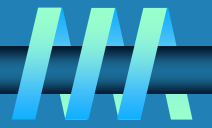
Lecture 14: Outline



- **Variable-Size-Decrease Algorithms**
  - **Median and Selection Problem (cont.)**
    - **QuickSelect**
  - **Interpolation Search**
  - **Binary Search Tree Operations**
  - **Game of Nim**

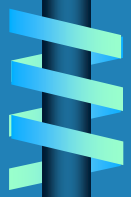


Variable-Size-Decrease Algorithms



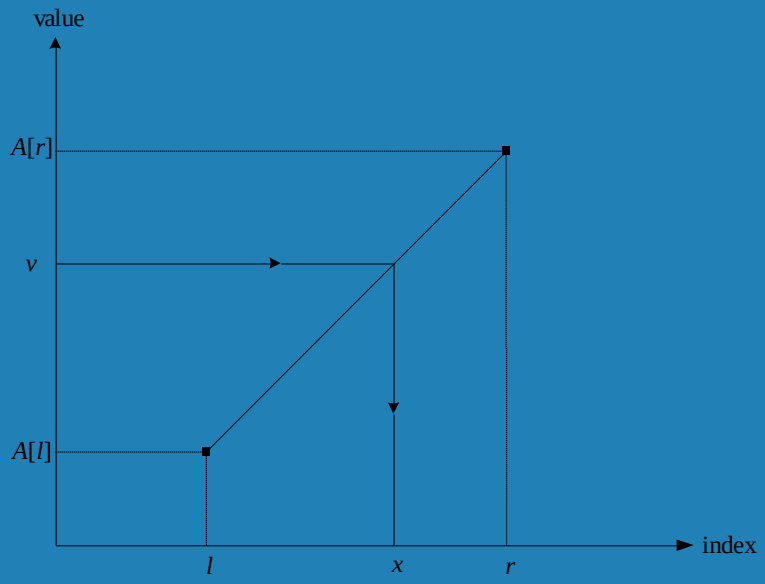
- In the variable-size-decrease variation of decrease-and-conquer, instance size reduction varies from one iteration to another

- Examples:
  - Euclid's algorithm for greatest common divisor
  - partition-based algorithm for selection problem
  - interpolation search
  - some algorithms on binary search trees
  - Nim and Nim-like games



Interpolation Search (bonus/optional topic)

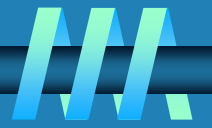
- Searches a sorted array similar to binary search but estimates location of the search key in A[l..r] by using its value v.
- Specifically, the values of the array's elements are assumed to grow linearly from A[l] to A[r]



- The location of v is estimated as the x-coordinate of the point on the straight line through (l, A[l]) and (r, A[r]) whose y-coordinate is v:

$$x = l + \lfloor (v - A[l])(r - l)/(A[r] - A[l]) \rfloor$$

Analysis of Interpolation Search



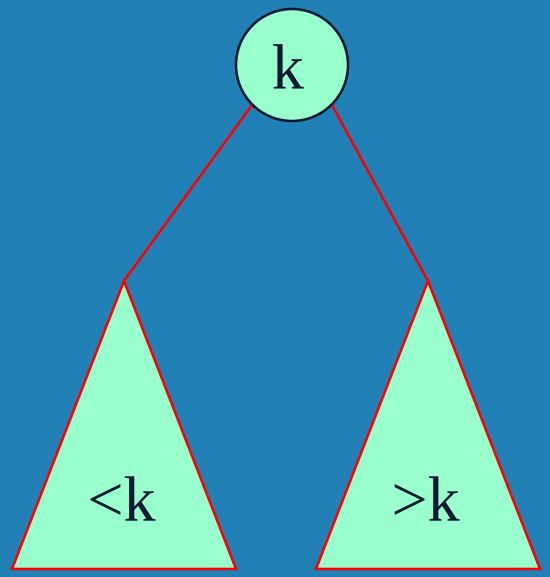
- Efficiency
- average case: C(n) < log2 log2 n + 1
- worst case: C(n) = n

- Preferable to binary search only for VERY large arrays and/or expensive comparisons

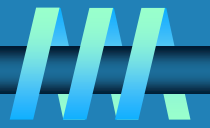
- Has a counterpart, the method of false position (regula falsi), for solving equations in one unknown (Sec. 12.4)

Binary Search Tree Algorithms

- Several algorithms on BST requires recursive processing of just one of its subtrees, e.g.,
  - Searching
  - Insertion of a new key
  - Finding the smallest (or the largest) key



Searching in Binary Search Tree

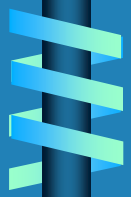


Algorithm BTS(x, v)

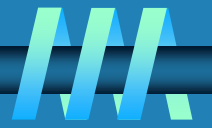
//Searches for node with key equal to v in BST rooted at node x

```
if x = NIL  return -1
else if  v = K(x)  return x
else if  v < K(x)  return BTS(left(x), v)
else return BTS(right(x), v)
```

- **Efficiency**
- **worst case: C(n) = n**
- **average case: C(n) ≈ 2ln n ≈ 1.39log2 n**

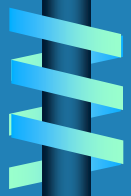


One-Pile Nim

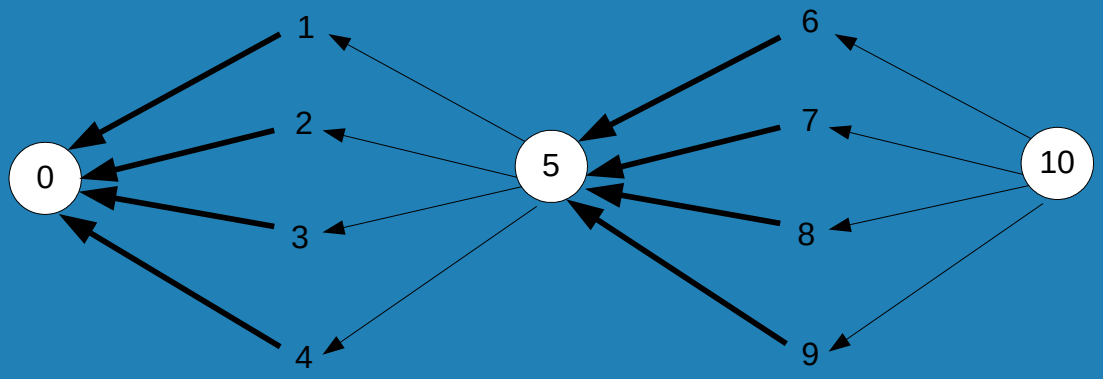


- There is a pile of n chips. Two players take turn by removing from the pile at least 1 and at most m chips. (The number of chips taken can vary from move to move.) The winner is the player that takes the last chip.
- Who wins the game – the player moving first or second, if both player make the best moves possible?

- It's a good idea to analyze this and similar games "backwards", i.e., starting with n = 0, 1, 2, …



Partial Graph of One-Pile Nim with m = 4



Vertex numbers indicate *n*, the number of chips in the pile.

***m*= 4**

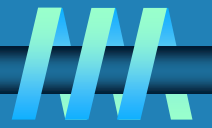
The losing position for the player to move are circled. Only winning moves from a winning position are shown (in bold).

One-Pile Nim with m = 4

- **Winning positions:**
  - $1 \leq n \leq m$
  - $m + 2 \leq n \leq 2m + 1$

- **Generalization:**
  - The player moving first wins iff n is not a multiple of 5 (more generally, m+1);
  - the winning move is to take n mod 5 (n mod (m+1)) chips on every move.

N-pile Nim



- Two players take turns removing objects from heaps, which are rows in this case.
- On each turn, you can remove as many coins as you (including all of them) from one pile.
- The player that makes the last move wins.
  - Or to think from a different perspective, the player that cannot make another move loses.