# On Implicit Notions in Algorithmic Problem Solving

David Ginat
Tel-Aviv University

# Majority in a very long list

Input: very long list L, of long-integers

Output: value that appears a majority number of times, if there is one

Limitations: the input cannot be all kept in memory, but may be read a few times

[Boyer & Moor, 1991]

# Intuitive Approaches

Count value appearances

But … requires too many counters

Narrow down the range of potential values

But … requires many iterations

# A Different Perspective

Declarative observation: if v is majority in L then v is also majority in L minus v and u

Operative implementation: repeatedly delete pairs of different values; the remaining value will be a sole candidate for majority

The candidate will be checked for majority

# The Notion of “Candidate”

Appears in: max computation, linear searches

Basic idea: “… start with something, improve and re-improve …”

Additional ways of flexible utilizations

Here: first - seek a candidate,
then – validate that it is satisfying

# Problem Solving Toolbox

Design Patterns – generic schemes, e.g. sum

Data Structures – queue, stack, trees, …

Design Techniques – dyn-prog, div-&-conq, …

But also implicit notions: complement, symmetry, pigeon-hole, invariance, candidate, and more …

# Implicit Notions

Very useful

Of various forms

But – soft, their utilization requires flexibility, abstraction, declarative perspective

And they not underlined in textbooks and teaching materials

# Longest Plateau

Input: long list L of integers

Output: longest sub-sequence in which every two elements differ by at most 1

e.g.: 3 4 4 3 5 5 4 3 3 2 3 2 2 1 2

# Common Approaches

1. Compute the max-min diff in every sub-list.
2. Handle overlapping plateaus, by "keeping history" and "tracing points of change".

e.g.: 3 4 4 3 5 5 4 3 3 2 3 2 2 1 2

Approach #1 is inefficient

Approach #2 is cumbersome and error-prone

# “Candidate” Approach

Every input value Belongs to

two concurrent candidate plateaus

The next input value may:

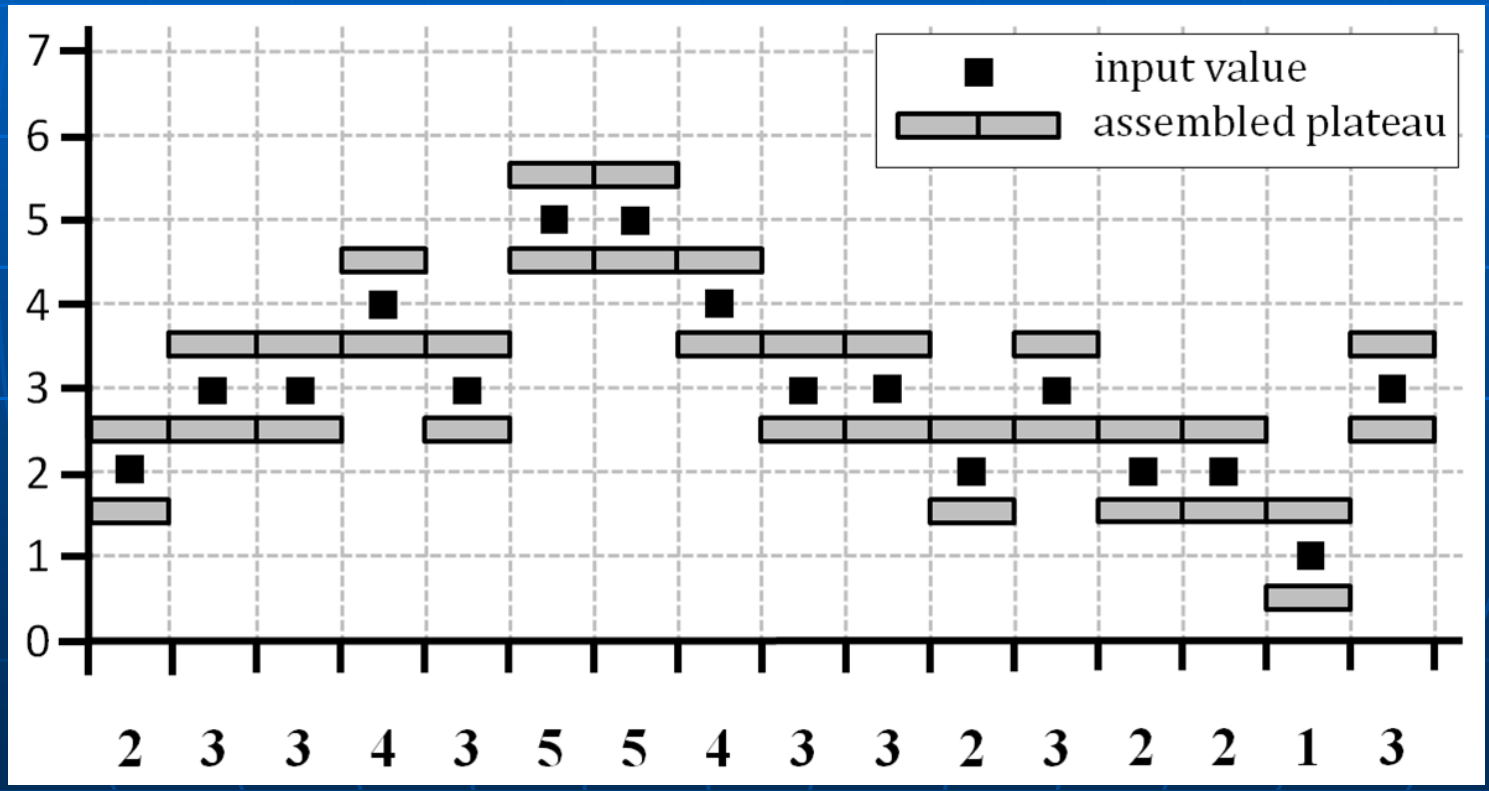
Extend a plateau, or

End a plateau and start another

An Abstract point of view

# “Candidate” Approach

At any time: two concurrent candidate plateaus



# “Candidate” Utilizations

1. Start with an init candidate, and replace it repeatedly (the natural tendency).

2. Seek a sole candidate, and validate it
(in: Finding Majority, Finding Celebrity, …).

3. Progress with concurrent candidates (from which one will be chosen).

4. Additional, flexible variations.

# Safari

N animals, every two may "get along", or not

Exactly 2/3 of the N animals get along

Input: all the "get along" pairs of animals

Output: 1/3 of the N animals,
that can be put together

[ reference: … ]

# Natural Approaches

Graph G: each edge is a "get-along" edge

Given: there is a clique CL of size 2N/3 in G

Find a clique of size N/3 (or larger)

Various attempts of (local) optimizations of brute-force constructions of the clique

# A Different Perspective

Declarative observation-1: an edge e in G-complement connects two animals that at least one of them is not in CL.

Declarative observation-2: if we remove the two animals incident on e then we remove at most one animal of CL.

Declarative observation-3: N/3, or less, removals will yield a subset of CL, of size N/3 or larger

# Computational Thinking

Abstraction, Flexibility, and more

Declarative and Operative perspectives

Not only with design patterns, data structures, and design techniques

But also with implicit notions

Should be underlined and practiced explicitly in a variety of flexible utilizations