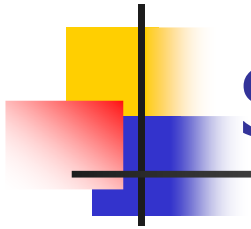
Search Algorithms

Paolo Camurati

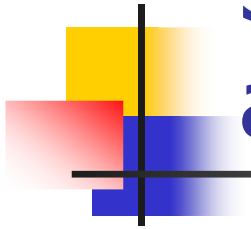
Dip. Automatica e Informatica
Politecnico di Torino



Search algorithms on arrays

Search

- Problem definizion
  - Is key k present in array v[N]?
  - Yes/No
- Input: v[N], k
- Output: Yes/No, if Yes, where in the array (index in the array)

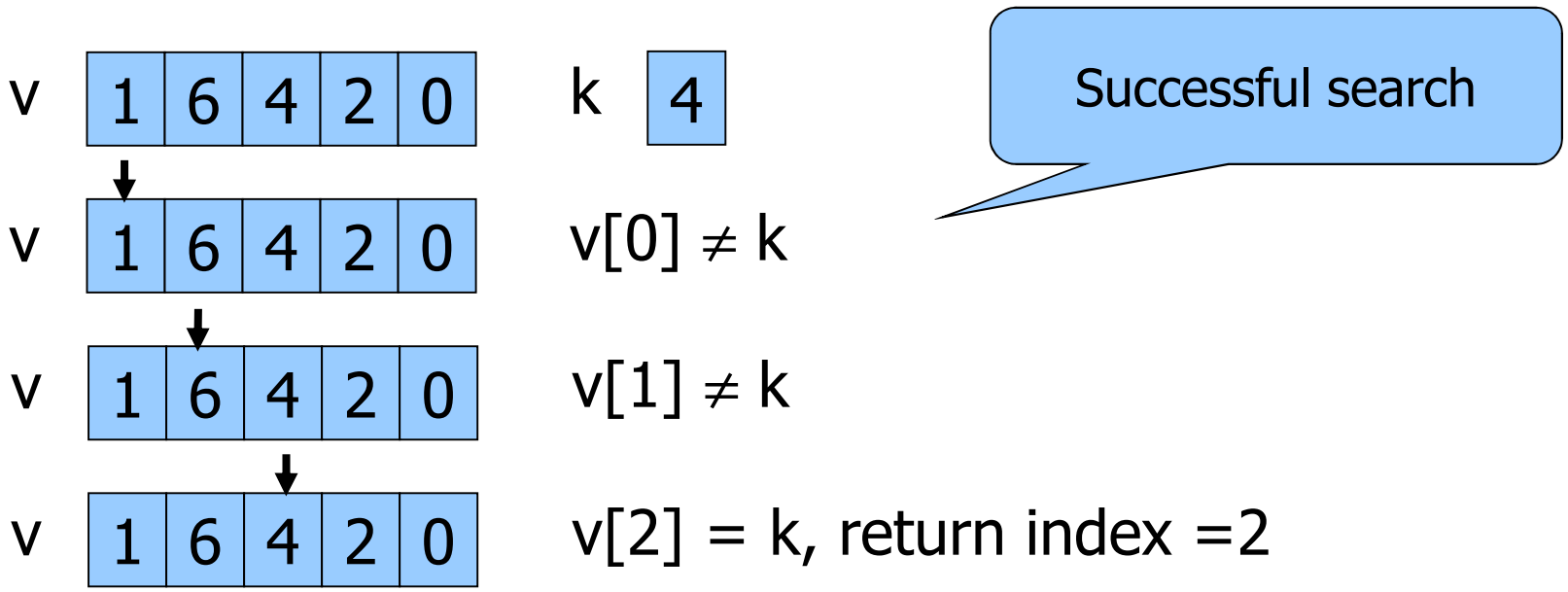


Steps to developing a usable algorithm

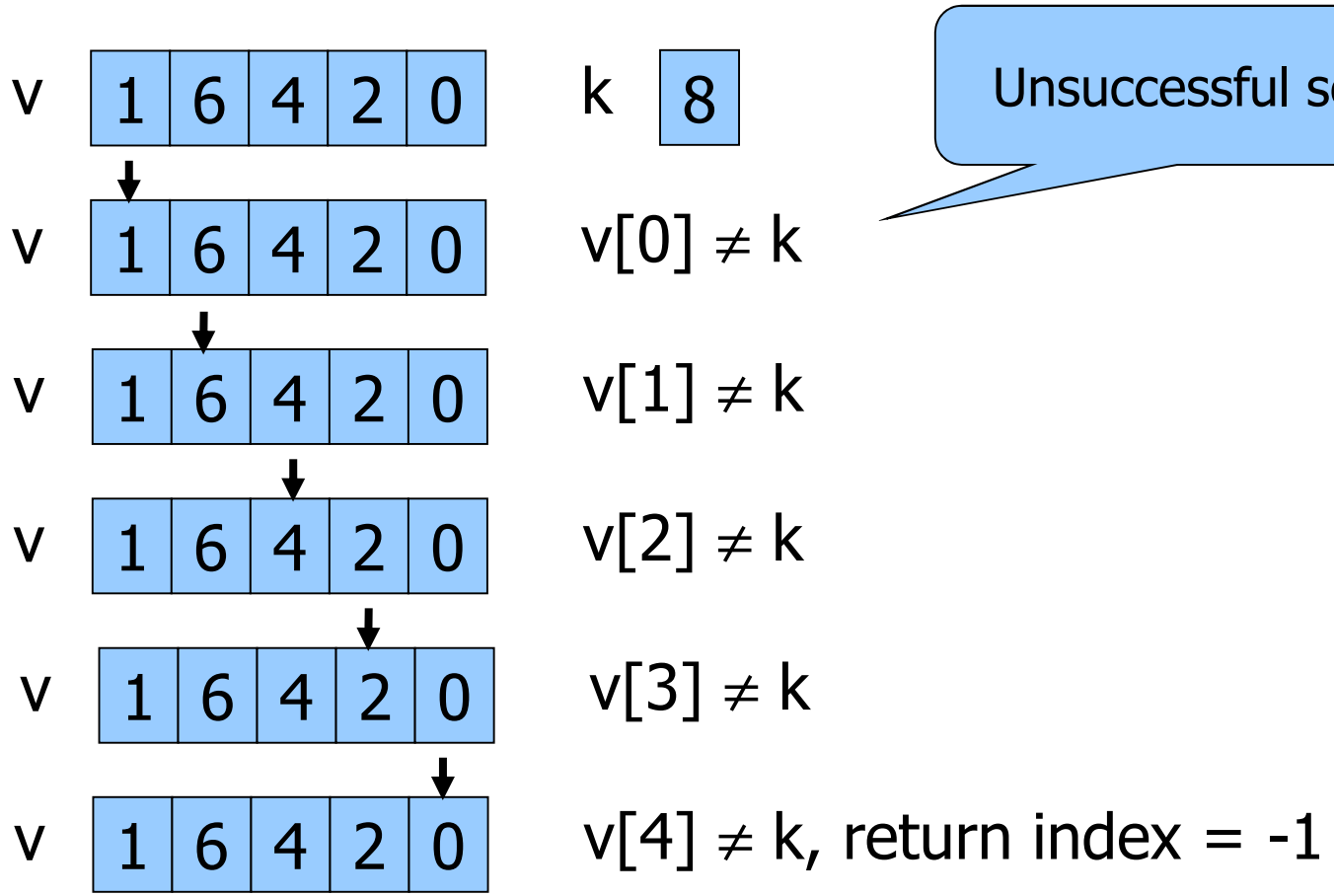
- The scientific method
  - Model the problem
  - Find an algorithms to solve it
  - Fast enough? Fits in memory?
  - If not, figure out why
  - Find a way to address the problem
  - Iterate until satisfied

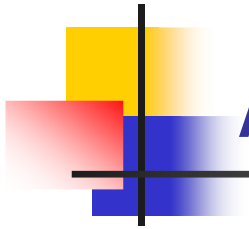
Algorithm 1: Sequential search

Sequential search: scan the array from the first element to potentially the last one, comparing key k and current value



Algorithm 1: Sequential search

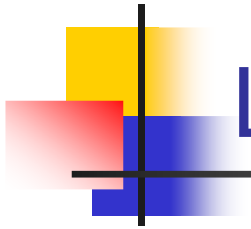




Algorithm 1: Sequential search

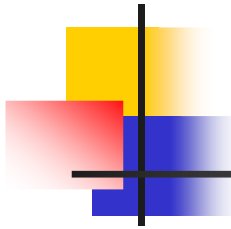
```
int LinearSearch (int v[], int l, int r, int k) {
  int i = l;
  int found = 0;

  while (i <= r && found == 0) {
    if (k == v[i]) {
      found = 1;
    } else {
      i++;
    }
  }
  if (found ==0)
    return -1;
  else
    return i;
}
```



Linear Search: Complexity Analysis

- We consider n numbers for a search miss and in average n/2 for a search hit
- T(n) grows linearly with n



Algorithm 2: Binary search

Search in Sorted Arrays

Binary search

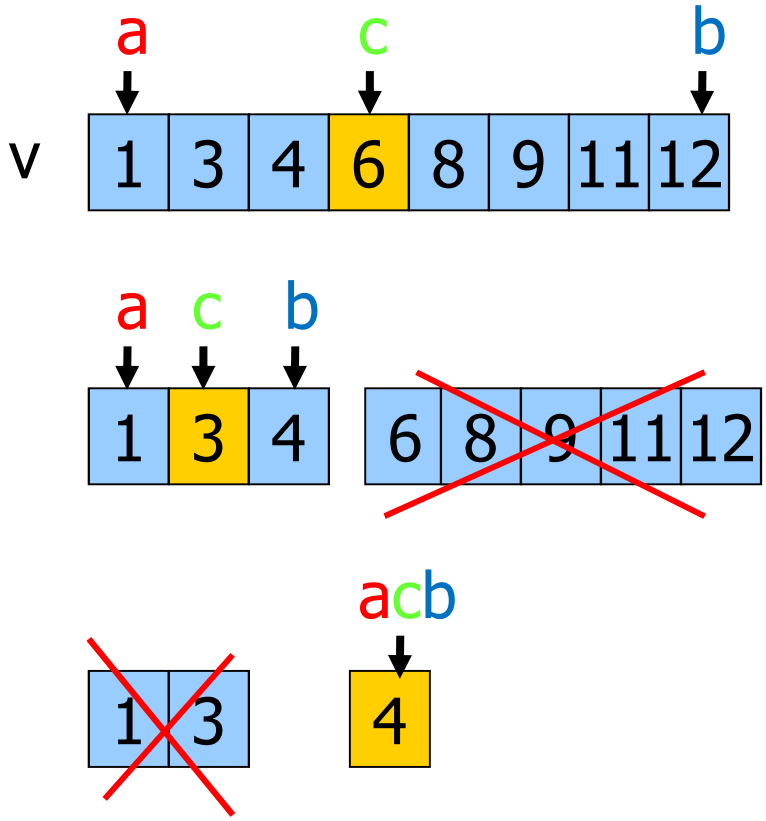
Problem definition

- Given a sorted array v[N]
- Is key k present in v[N]?
- Yes/No

Approach

- At each step
  - compare k with middle element in the array
    - =: termination with success
    - <: search continues on left subarray
    - >: search continues on right subarray

Algorithm 2: Binary search

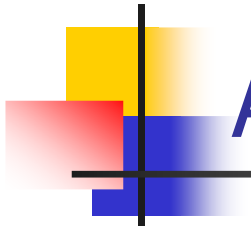


k 4

y = middle element
a = leftmost array index
b = rightmost array index
c = index of middle element

Seach hit

v[2] = k, return index =2

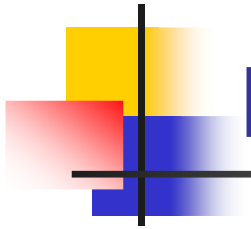


Algorithm 2: Binary search

```
int BinSearch (int v[], int a, int b, int k) {
  int c;

  while(a <= b){
    c = (a+b) / 2;

    if(v[c] == k) {
      return(c);
    }
    if(v[c] < k) {
      a = c+1;
    } else {
      b = c-1;
    }
  }
  return(-1);
}
```



Binary Search: Complexity Analysis

- The array to be examined
  - At the beginning contains n numbers
  - At the 2nd iteration contains about n/2 numbers
  - …
  - At the i-th iteration contains about $n/2^i$ numbers
- Termination occurs when the array to be examined contains 1 number
  - thus $n/2^i = 1 \rightarrow n = 2^i \rightarrow i = \log_2(n)$
- T(n) grows logarithmically with n