Fakultas Informatika
School of Computing
Telkom University

CSG2A3
ALGORITMA dan STRUKTUR DATA

Searching and Sorting on

Single Linked List

Sequential Search with Sentinel

- Supposedly, sequential search using sentinel is a searching algorithm aimed to reduce the complexity which is more suitable for linked list
  - Rather than using it on array

- Sequential search

  what happens each time through the loop:

1) compare *array[i]* against *key*
2) increment counter *i* and compared against *n*

Sequential Search with Sentinel

- Sequential search
  what happens each time through the loop:

1) compare *array[i]* against *key*
2) increment counter *i* and compared against *n*

- There are 2 comparison processes inside the loop
- The aim is to eliminate one of those comparisons
  - So the loop might be a lot faster

Sequential Search with Sentinel

- Sequential search
  what happens each time through the loop:

1) compare *array[i]* against *key*
2) increment counter *i* and compared against *n*

- We need step (2) because:
  - We might run off the end of array[] which might causing undefined behavior,
  - Obviously because we aren't sure that key is in array[].
- If we knew that the key was in array[], then we could skip step (2)

Sequential Search with Sentinel

- We can ensure that the item we're looking for is in the array.
  - By putting a copy of it at the end of the array
  - This copy is called a *sentinel*
  - Eliminate process of comparing counter *i* against *n*
- The application is better using pointer
  - Eliminate process of increment counter *i*

Sequential Search with Sentinel

- Sequential search using sentinel

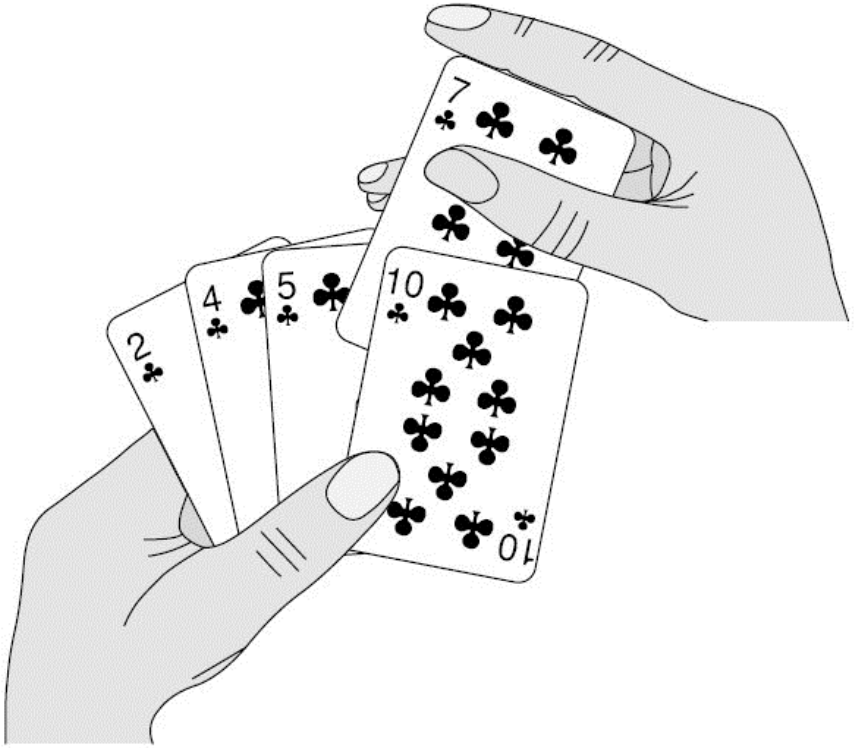
  what happens each time through the loop:

1) compare *info(P)* against *key*
2) next(P)

Exercise

- Assumed that the insert functions (first, after, last) are already defined, write a procedure algorithm for sequential search using sentinel for single linked list

Insertion Sort



Insertion Sort

- An Efficient sorting algorithm for (quite) small data sets
  - $O(n^2)$ comparisons

- Insertion sort on array

  what happens each time through the loop:

1) Go through the elements in the unsorted part sequentially
2) Insert each element into the sorted part by searching for it's correct position

Insertion Sort

- Insertion sort on array

  what happens each time through the loop:

1) Go through the elements in the unsorted part sequentially
2) Insert each element into the sorted part by searching for it's correct position

- Step (2) on array costs O(n) since there is the possibility of swapping process for each insertion
- Therefore the cost of the entire algorithm is $O(n^2)$

Nb: you'll learn more about this next year

Insertion Sort

- Insertion sort on array

  what happens each time through the loop:

1) Go through the elements in the unsorted part sequentially
2) Insert each element into the sorted part by searching for it's correct position

- Using linked list, there is no swapping process in insertion
  - Cost of step (2) is O(1)
- Therefore the cost of the entire algorithm became O(n)

Question?

Fakultas Informatika
School of Computing
Telkom University

THANK YOU