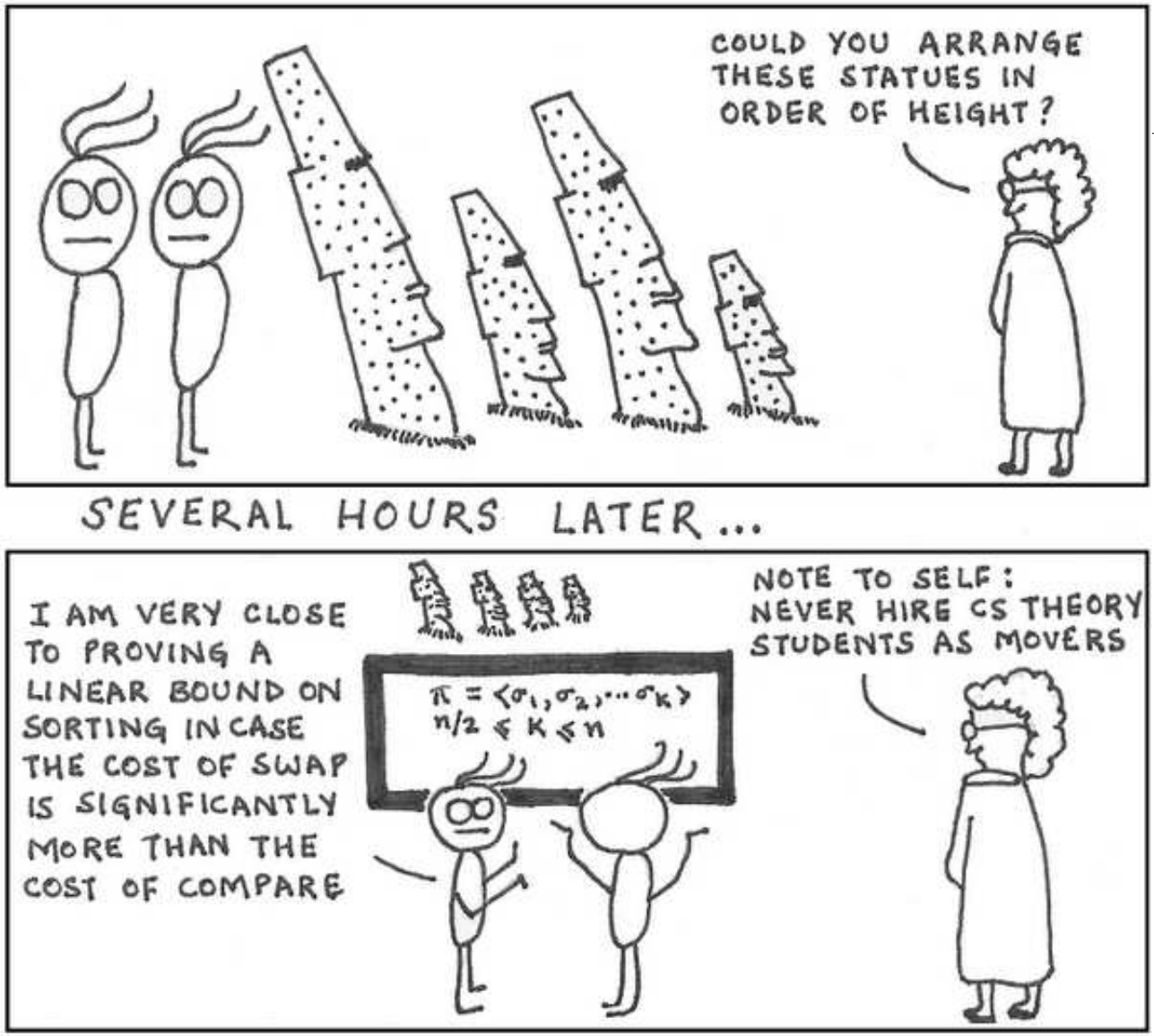


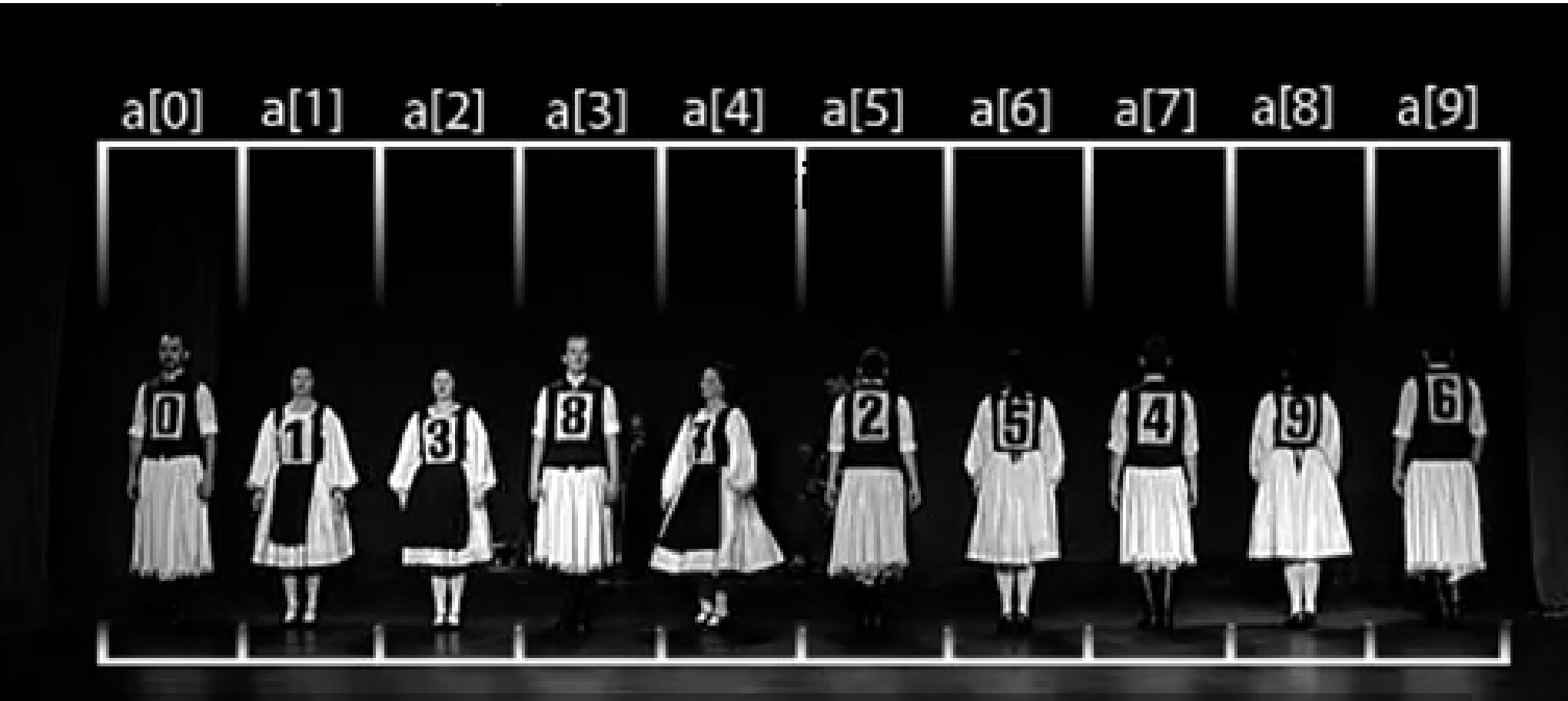
Slides #20

Sorting

Topics

1) How can we sort data in an array?
    a) Selection Sort
    b) Insertion Sort

Ch11: p436 – p39
http://www.youtube.com/watch?v=ROalU379l3U



Sorting

- Sorting is the process of..

- Examples:
  - Sorting an array of names into alphabetical order.
  - Sorting an array of stock prices into descending order.
- It's a classic computer science problem:
  - Theoretical analysis possible (later).
  - Many possible sorting algorithms.
  - Generally algorithms evaluated...

Selection sort

- Algorithm Idea:
  - Search list to find the...
    - Exchange first element with minimum.
  - Search list to find the...
    - Exchange second element with next smallest.
  - ... and so on.
  - Repeat until all items are in their place.

Selection sort example

- Sort this list using selection sort:
  8 1 6 9 6 4 2 0
- 0 1 6 9 6 4 2 8
- 0 1 6 9 6 4 2 8
- 0 1 2 9 6 4 6 8
- 0 1 2 4 6 9 6 8
- 0 1 2 4 6 9 6 8
- 0 1 2 4 6 6 9 8
- 0 1 2 4 6 6 8 9

Selection sort

```
// Find the index of the smallest remaining element
int  findSmallest(int data[], int size, int startingAt);
void swap(int &x, int &y);

void selectionSort(int data[], int size)
{
    for (int i = 0; i < size-1; i++) {
        int idxSmallest = findSmallest(data, size, i);
        swap(data[idxSmallest], data[i]);
    }
}


int main() {
    const int POINTS = 5;
    int sortMe[] {5, 10, 1, 18, 3};

    selectionSort(sortMe, POINTS);
    ...
}
```

Sorting:
Insertion Sort

Insertion sort

- Insertion Sort functions by:
  Repeatedly inserting the next number from the list into an..

- Algorithm description:
  - Skip the 1st element; it's already a sorted sub-list!
  - Take the 2nd element, insert it into the sorted sub-list.
  - Take the 3rd element, insert it into the sorted sub-list.
  - ...
  - Repeat until...
    has been inserted into the sorted sub-list.

Insertion sort example

- Sort this list using insertion sort:
  8 1 6 9 6 4 2 0
- 1 8 6 9 6 4 2 0
- 1 6 8 9 6 4 2 0
- 1 6 8 9 6 4 2 0
- 1 6 6 8 9 4 2 0
- 1 4 6 6 8 9 2 0
- 1 2 4 6 6 8 9 0
- 0 1 2 4 6 6 8 9

Insertion sort

```
void insertionSort (int data[], int size) {
    for (int i = 1; i < size; i++) {
        // Grab next element to insert into sorted list
        int item = data[i];

        // Make a hole by shifting bigger values right
        int holeIdx = i;
        for (holeIdx = i; holeIdx > 0; holeIdx--) {
            if (data[holeIdx - 1] > item) {
                data[holeIdx] = data[holeIdx - 1];
            } else {
                // The hole is in the right spot!
                break;
            }
        }

        // Put the item into this hole
        data[holeIdx] = item;
    }
}
```

```
int main() {
    const int SIZE = 5;
    int sortMe[] {5, 10, 1, 18, 3};

    insertionSort(sortMe, SIZE);
    ...
}
```

Criteria for selecting a sort algorithm

- Simplicity:
  Simple algorithms are easier to...

- Faster algorithms generally win out for..
  - Ex: all SFU students, all Canadians seniors.
    - # Item Comparisons
    - # Item Swaps

- How much memory is needed for each algorithm?
- Some sort algorithms use large amounts of memory.

Review

- Which sort algorithm most resembles sorting a hand of cards as you are dealt cards one at a time?

- Draw out sorting the following using each sort. Show only the swaps, and what is already sorted.

  4 8 1 0 7

Summary

- Searching and Sorting are two classic computing science problems.
- Searching:
  - Linear: Look at each element to find item.
  - Binary: Look half way through sorted list to find which half target element could be in.
- Sorting:
  - Selection sort: Finds next smallest item.
  - Insertion sort: Sort next item into existing list.
- Runtime efficiency (time) is how most algorithms are characterized.