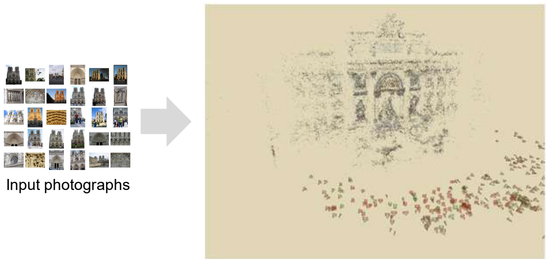
## COMP20003 Algorithms and Data Structures Why sorting?

Kris Ehinger
Department of Computing and Information Systems
University of Melbourne
Semester 2

---

## Sorting example




N. Snavely, S. M. Seitz, R. Szeliski, "Photo tourism: Exploring photo collections in 3D," ACM SIGGRAPH, 2006.
http://phototour.cs.Washington.edu



---

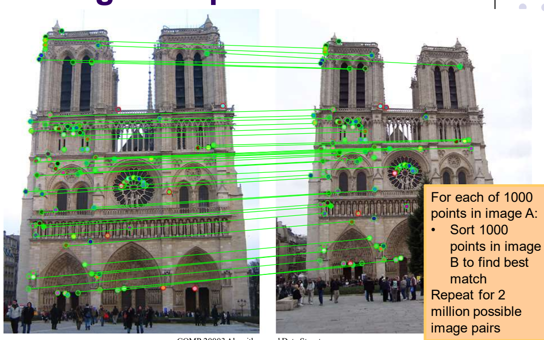
## Sorting example


N. Snavely, R. Garg, S. M. Seitz, R. Szeliski, "Finding paths through the world's photos," ACM SIGGRAPH, 2008.
http://phototour.cs.Washington.edu/findingpaths



---

## Sorting example



For each of 1000 points in image A:
- Sort 1000 points in image B to find best match

Repeat for 2 million possible image pairs



Figure: J. Hays

## Why is sorting useful to study?

- Sorting has many applications and is used widely
  - In the business world
  - In science
  - and many other disciplines
- Sorting is used within many *other* algorithms
  - very well-studied
  - demonstrates fundamental concepts CS
- Skiena: Chapter 4



## Why is sorting useful to study?

- Different algorithms for sorting have different properties, which affect performance

| $n$ | $n^2/4$ | $n \lg n$ |
|---|---|---|
| 10 | *25* | 33 |
| 100 | 2,500 | *664* |
| 1,000 | 250,000 | *9,965* |
| 10,000 | 25,000,000 | *132,877* |
| 100,000 | 2,500,000,000 | *1,660,960* |

Table from Skiena, The Algorithm Design Manual

- When data are big, efficiency matters, again!



## Selection Sort

[4, 3, 5, 1, 2]

[1, 3, 5, 4, 2]

[1, 2, 5, 4, 3]

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]



## Selection Sort

```
void selection(item* A, int n)
{
   int i,j,min;
   for( i = 0; i < n-1; i++ )      /* why n-1? */
   {
       min = i;
       for( j = i+1; j < n; j++ )
       {
           if( cmp( A[j], A[min] ) < 0 ) min = j;
       }
        SWAP( A[i], A[min] );
   }
}
```

https://www.jdoodle.com/a/5uP

## Selection Sort

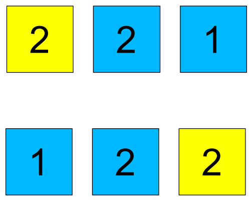
- Worst case:
- Best case:
- Average case:

- Why use it?
  - Useful when moving records is costly – selection sort requires O(n) swaps



## Selection Sort

- Is selection sort stable?






## Insertion Sort

[4, 3, 5, 1, 2]

[3, 4, 5, 1, 2]

[3, 4, 5, 1, 2]

[1, 3, 4, 5, 2]

[1, 2, 3, 4, 5]



## Insertion Sort: The idea

```
void insertion(item* A, int n)
{
   int i,j,val;
   for( i=1; i < n; i++ )
   {
       val = A[i]; j=i;
       while( A[j-1] > val )
       {
           A[j] = A[j-1]; j--;
       }
       A[j] = val;
   }
} /* this code doesn't usually work - why not? */
```

https://www.jdoodle.com/a/5uQ

## Insertion Sort

- In order to fix it, you need to either:



## Insertion Sort

- Worst case:
- Average case:
- Best case:
- Stability?

- Why use it?



## Sorting demos

Animations:
https://people.ok.ubc.ca/ylucet/DS/ComparisonSort.html
https://www.toptal.com/developers/sorting-algorithms

The sound of sorting:
https://www.youtube.com/watch?v=t8g-iYGHpEA



## Divide and Conquer

- Divide-and-conquer is a common strategy in efficient algorithms

- Divide and Conquer Strategy:
  - Divide instance of problem into smaller instances
  - Solve smaller instances – usually recursively
  - *e.g.* Binary Search

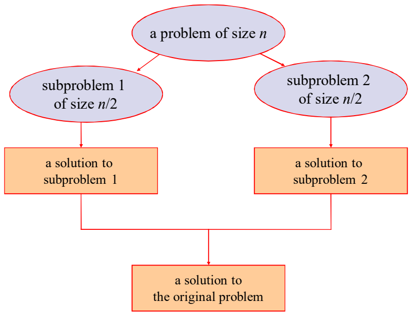
## Divide and Conquer

In sorting, the usual strategy is:

- Divide instance of problem into smaller instances
- Solve smaller instances – usually recursively
- Combine smaller solutions



## Divide and Conquer, or Split-solve-join






## Split-solve-join

- Hard split, easy join: Quicksort
- Easy split, hard join: Mergesort