Sorting & Shortest Path

Teaching Demonstration

Tashfeen, Ahmad

Petree College of Arts & Sciences, Oklahoma City University

Interview Presentation, Summer 2023

Oklahoma City
UNIVERSITY

Overview

Introduction

1. Doctoral Candidate at University of Oklahoma.
2. Research in Cryptography (factoring and post-quantum models) and Artificial Intelligence.
3. Have taught classes, e. g., Intro. to Java/C++/C & Discrete Math(s).
4. I have a Scottish Fold cat, Romeo and two lemon trees I grew from seed.

Introduction

Figure 1: Romeo and my lemon trees.

Sorting Problem

1. Let $A \subset \mathbb{R}$ be a set of n real numbers, find $\{1 \leq i \leq n : x_i\}$ such that,

$$i < j \Rightarrow x_i < x_j \quad \text{take} \quad \{2, 1.5, -\pi\} \rightarrow \{-\pi, 1.5, 2\}$$

2. Why do we care?
    1. We sort hands in card games to quickly respond to bets.
    2. How can you establish majority in unsorted and sorted data?

Figure 2: A hand of Bridge.

Bubble Sort

1. The main idea is, find the largest number and put it at the end. *How?*

$$
\begin{array}{ccccc}
\{\boxed{3,2}\} & \rightarrow & \{\boxed{2,3}\} & & \\
\{\boxed{3,2},1\} & \rightarrow & \{2,\boxed{3,\ 1}\} & \rightarrow & \{2,\boxed{1,\ 3}\} \\
\{\boxed{2,1},3\} & \rightarrow & \{\boxed{1,2},3\} & \rightarrow & \{1,\boxed{2,\ 3}\} \\
\{\boxed{1,2},3\} & \rightarrow & \{1,\boxed{2,\ 3}\} & \rightarrow & \text{No Swaps!}
\end{array}
$$

2. Hence the name, "Bubble" sort.

$$\boxed{\text{Bubble}}\ \text{Sort}$$

Bubble Sort

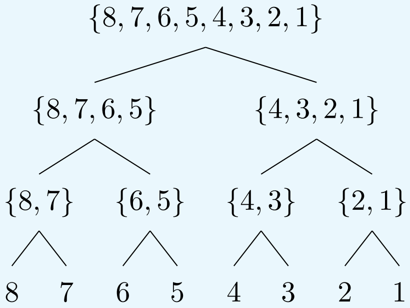
```python
def bubble_sort(ls, k, swapped=True):
    if not swapped:
        return ls
    swapped = False
    for i in range(len(ls)-1):
        if ls[i] > ls[i+1]:
            ls[i], ls[i+1] = ls[i+1], ls[i]
            swapped = True
        print(f'{k:2d}. {ls}')
        k += 1
    return bubble_sort(ls, k, swapped)

bubble_sort([5, 4, 3, 2, 1], 1)
```

Listing 1: Bubble sort in Python.

```
 1. [4, 5, 3, 2, 1]
 2. [4, 3, 5, 2, 1]
 3. [4, 3, 2, 5, 1]
 4. [4, 3, 2, 1, 5]
 5. [3, 4, 2, 1, 5]
 6. [3, 2, 4, 1, 5]
 7. [3, 2, 1, 4, 5]
[snip]
14. [1, 2, 3, 4, 5]
15. [1, 2, 3, 4, 5]
16. [1, 2, 3, 4, 5]
17. [1, 2, 3, 4, 5]
18. [1, 2, 3, 4, 5]
19. [1, 2, 3, 4, 5]
20. [1, 2, 3, 4, 5]
```

Listing 2: Output for $\{5, 4, 3, 2, 1\}$.

1. The main idea is, *divide and conquer.*



2. Now we merge from the bottom to the top, hence “Merge” sort.

Merge Sort

- Level 4

$$\{8, 7, 6, 5, 4, 3, 2, 1\}$$

$$\{8, 7, 6, 5\} \qquad \{4, 3, 2, 1\}$$

$$\{8, 7\} \qquad \{6, 5\} \qquad \{4, 3\} \qquad \{2, 1\}$$

$$8 \quad 7 \quad 6 \quad 5 \quad 4 \quad 3 \quad 2 \quad 1$$

Merge Sort

- Level 4

$$\{8, 7, 6, 5, 4, 3, 2, 1\}$$

$$\{8, 7, 6, 5\} \qquad \{4, 3, 2, 1\}$$

$$\{8, 7\} \quad \{6, 5\} \quad \{4, 3\} \quad \{2, 1\}$$

$$7 \quad 8 \quad 5 \quad 6 \quad 3 \quad 4 \quad 1 \quad 2$$

Merge Sort

- Level 3

$$\{8, 7, 6, 5, 4, 3, 2, 1\}$$

$$\{8, 7, 6, 5\} \qquad \{4, 3, 2, 1\}$$

$$\{7, 8\} \qquad \{5, 6\} \qquad \{3, 4\} \qquad \{1, 2\}$$

$$7 \quad 8 \quad 5 \quad 6 \quad 3 \quad 4 \quad 1 \quad 2$$

Merge Sort

- Level 2

$$\{8, 7, 6, 5, 4, 3, 2, 1\}$$

$$\{5, 6, 7, 8\} \qquad \{1, 2, 3, 4\}$$

$$\{7, 8\} \quad \{5, 6\} \quad \{3, 4\} \quad \{1, 2\}$$

$$7 \quad 8 \quad 5 \quad 6 \quad 3 \quad 4 \quad 1 \quad 2$$

Merge Sort

- Level 1

$$\{1,2,3,4,5,6,7,8\}$$

$$\{5,6,7,8\} \qquad \{1,2,3,4\}$$

$$\{7,8\} \quad \{5,6\} \quad \{3,4\} \quad \{1,2\}$$

$$7 \quad 8 \quad 5 \quad 6 \quad 3 \quad 4 \quad 1 \quad 2$$

Merge Sort

```python
def merge_sort(ls):
    if len(ls) == 1:
        return ls
    left = merge_sort(ls[:len(ls)//2])
    right = merge_sort(ls[len(ls)//2:])
    return left + right if left[0] < right[0] else right + left

print(merge_sort([8,7,6,5,4,3,2,1]))
```

Listing 3: Merge sort in Python.

We get: $[1, 2, 3, 4, 5, 6, 7, 8]$

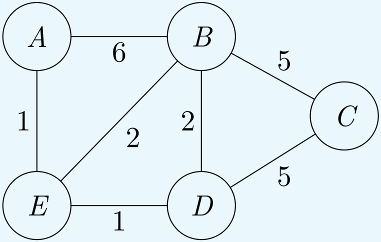
Dijkstra's Algorithm

- Greedy path finding algorithm.
- Used everywhere, e. g., Networks, Game Theory & AI.

Figure 3: Edsger Wybe Dijkstra (Dutch Mathematician)

Dijkstra's Algorithm

- Graph



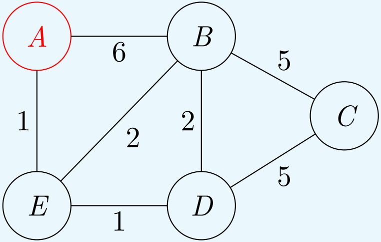
- Table

To	Weight	From
A	2^{100}	?
B	2^{100}	?
C	2^{100}	?
D	2^{100}	?
E	2^{100}	?

Table 1: Routing Table

Dijkstra's Algorithm

- Graph



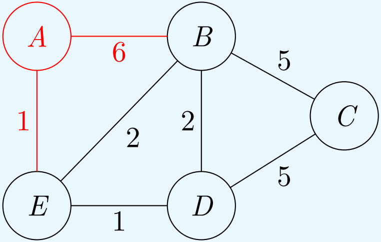
- Table

To	Weight	From
A	0	A
B	2^{100}	?
C	2^{100}	?
D	2^{100}	?
E	2^{100}	?

Table 2: Routing Table

Dijkstra's Algorithm

- Graph



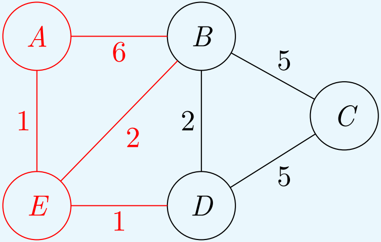
- Table

To	Weight	From
A	0	A
B	6	A
C	2^{100}	?
D	2^{100}	?
E	1	A

Table 3: Routing Table

Dijkstra's Algorithm

- Graph



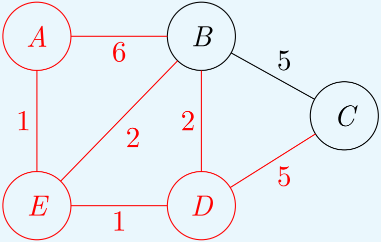
- Table

To	Weight	From
A	0	A
B	3	E
C	2^{100}	?
D	2	E
E	1	A

Table 4: Routing Table

Dijkstra's Algorithm

- Graph



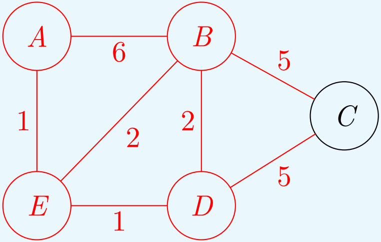
- Table

To	Weight	From
A	0	A
B	3	E
C	7	D
D	2	E
E	1	A

Table 5: Routing Table

Dijkstra's Algorithm

- Graph



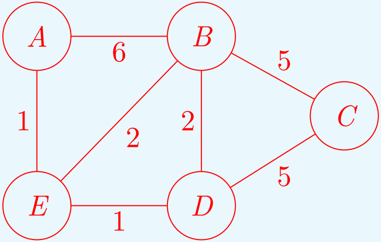
- Table

To	Weight	From
A	0	A
B	3	E
C	7	D
D	2	E
E	1	A

Table 6: Routing Table

Dijkstra's Algorithm

- Graph



- Table

To	Weight	From
A	0	A
B	3	E
C	7	D
D	2	E
E	1	A

Table 7: Routing Table

Dijkstra's Algorithm

Correctness Why does the algorithm work?
At each step, the routing table always contains the shortest paths for the seen vertices. I. e., the greedy approach works.

Complexity We must find the minimum weighing vertex at each step. let v be the number of vertices then,

$$\mathcal{O}(v \lg(v))$$

We must decrease the weight of each edge at least once. Let e be the number of edges.

$$\mathcal{O}(e \lg(v))$$

Combined,

$$\mathcal{O}((e+v)\lg(v))$$

Or, if the graph is minimally connected planar,

$$\mathcal{O}((v-1+v)\lg(v)) = \mathcal{O}(v \lg(v)) = \mathcal{O}(v^2)$$

Conclusion

We went over,

- Bubble sort.
- Merge sort.
- Dijkstra's path finding algorithm.

Thank You!
Questions?