```python
# setup
from IPython.core.display import display,HTML
display(HTML('<style>.prompt{width: 0px; min-width: 0px; visibility: collapse}</style>'))
display(HTML(open('rise.css').read()))

# imports
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(style="whitegrid", font_scale=1.5, rc={'figure.figsize':(12, 6)})
import time
```

CMPS 2200

Introduction to Algorithms

Recurrences - Intro & the Tree Method

Recurrences are a way to capture the behavior of recursive algorithms.

Key ingredients:

- Base case ($n = c$): constant time
- Inductive case ($n > c$): recurse on smaller instance and use output to compute solution

Actually recursion is a conceptual way to view algorithm execution, and we can reframe an algorithm specification to make it recursive.

In [2]:

```python
def selection_sort(L):
    for i in range(len(L)):
        print(L)
        m = L.index(min(L[i:]))
        L[i], L[m] = L[m], L[i]
    return L

selection_sort([2, 1, 4, 3, 9])
```

```
[2, 1, 4, 3, 9]
[1, 2, 4, 3, 9]
[1, 2, 4, 3, 9]
[1, 2, 3, 4, 9]
[1, 2, 3, 4, 9]
[1, 2, 3, 4, 9]
```

Out[2]:

In [3]:

```python
def selection_sort_recursive(L):
    print('L=%s' % L)
    if (len(L) == 1):
        return(L)
    else:
        m = L.index(min(L))
        L[0], L[m] = L[m], L[0]
        return [L[0]] + selection_sort_recursive(L[1:])
```

```
selection_sort_recursive([2, 1, 999, 4, 3])
```

```
L=[2, 1, 999, 4, 3]
L=[2, 999, 4, 3]
L=[999, 4, 3]
L=[4, 999]
L=[999]
[1, 2, 3, 4, 999]
```

Out[3]:

In [4]:
```python
# faster argmin.
L = [2, 1, 999, 4, 3]
m = min(enumerate(L), key=lambda x: x[1])[0]
```

Are these the same algorithm? Can we give a SPARC specification?

$$
\begin{aligned}
selectionsort\ &L = \\
&\text{if } |L| = 1 \text{ then} \\
&\quad \text{return } L \\
&\text{else} \\
&\quad \text{let} \\
&\quad\quad m = \text{minimum element in } L \\
&\quad \text{in} \\
&\quad\quad \text{Cons}(m, (selectionsort\ \langle x | x \in L\ and\ x \neq m \rangle))
\end{aligned}
$$

What is the running time and why?

$$
\begin{aligned}
W(n) &= W(n-1) + n & (1) \\
&= W(n-2) + 2n - 1 & (2) \\
&\ \ \vdots & (3) \\
W(n) &= \sum_{i=1}^{n} i & (4) \\
&= \frac{n(n+1)}{2} & (5) \\
&= \Theta(n^2). & (6)
\end{aligned}
$$

The recurrence for Selection Sort is somewhat simple - what if we have multiple recursive calls and split the input? (This is actually what *divide-and-conquer* algorithms do.)

We'll look at methods to solve recurrences in order to obtain big-O bounds for recursive algorithms.

We will:

- Get intuition for recurrences by looking the recursion tree.

- Develop the **brick** method to quickly state asymptotic bounds on a recurrence by looking at the shape of the tree.

Let's look at the specification and recurrence for Merge Sort:

$$mergeSort\ a =$$
$$\texttt{if } |a| \leq 1 \texttt{ then}$$
$$a$$
$$\texttt{else}$$
$$\texttt{let}$$
$$(l, r) = splitMid\ a$$
$$(l', r') = (mergeSort\ l \parallel mergeSort\ r)$$
$$\texttt{in}$$
$$merge(l', r')$$
$$\texttt{end}$$

Suppose that the merging step can be done with $O(n)$ work and $O(\log n)$ span. Then recurrence for the work is:

$$W(n) = \begin{cases} c_b, & \text{if } n = 1 \\ 2W(n/2) + c_1 n + c_2, & \text{otherwise} \end{cases}$$

How do we solve this recurrence to obtain $W(n) = O(n \log n)$?

alttext

The recursion tree for Merge Sort has linear work at every level except at the leaves. There are a logarithmic number of levels and a linear number of leaves so we obtain an asymptotic bound of $O(n \log n)$ for the work.

Solving Recurrences with the Tree Method

size at level i

cost at level i

merge-tree

Recipe:

1. Expand tree for two levels.
2. Determine the cost of each level i (i starts at 0).
3. Determine the number of levels
4. Cost = $\sum_{i=0}^{\text{num levels}}$ cost for level i
   - This last step usually involves using properties of series

E.g., for merge sort:

- level i contains 2^i nodes
- each node at level i costs $c_1 \frac{n}{2^i} + c_2$
- so, each level costs $2^i * (c_1 \frac{n}{2^i} + c_2) = c_1 n + 2^i c_2$
- since each level reduces size by half, we have $\lg n$ levels
- so, total cost of tree is:

$$W(n) = \sum_{i=0}^{\lg n} (c_1 n + 2^i c_2)$$

$$W(n) = \sum_{i=0}^{\lg n} (c_1 n + 2^i c_2)$$

To solve this, we'll make use of bounds for **geometric series**.

- For $\alpha > 1$: $\sum_{i=0}^{n} \alpha^i < \frac{\alpha}{\alpha-1} \cdot \alpha^n$

    - e.g., $\sum_{i=0}^{\lg n} 2^i < \frac{2}{1} * 2^{\lg n} = 2n$

- For $\alpha < 1$: $\sum_{i=0}^{\infty} \alpha^i < \frac{1}{1-\alpha}$

    - e.g., $\sum_{i=0}^{\lg n} \frac{1}{2^i} < 2$

plugging in...

$$= \sum_{i=0}^{\lg n} (c_1 n + 2^i c_2)$$

$$= \sum_{i=0}^{\lg n} c_1 n + \sum_{i=0}^{\lg n} 2^i c_2$$

$$= c_1 n \sum_{i=0}^{\lg n} 1 + c_2 \sum_{i=0}^{\lg n} 2^i$$

$$< c_1 n \lg n + 2 c_2 n$$

$$\in O(n \lg n)$$

What about the span?

tree

The recurrence for the span of Mergesort is:

$$S(n) = \begin{cases} c_3, & \text{if } n = 1 \\ S(n/2) + c_4 \lg n, & \text{otherwise} \end{cases}$$

Since each level of the recursion tree is concurrent and all nodes have the same cost, we have that

$$S(n) = \sum_{i=1}^{\lg n} \lg \frac{n}{2^i} \tag{7}$$

$$= \sum_{i=1}^{\lg n} (\lg n - i) \tag{8}$$

$$= \sum_{i=1}^{\lg n} (\lg n) - \sum_{i=1}^{\lg n} i \tag{9}$$

$$< \lg^2 n - \frac{1}{2}\lg n * (\lg n + 1) \ \left(\text{using } \sum_{i=1}^{n} = \frac{n(n+1)}{2}\right) \tag{10}$$

$$= \lg^2 n - \frac{1}{2}\lg^2 n - \frac{1}{2}\lg n \tag{11}$$

$$= O(\lg^2 n) \tag{12}$$

Another recurrence:

$$W(n) = \begin{cases} c_b, & \text{if } n = 1 \\ 2W(n/2) + n^2, & \text{otherwise} \end{cases}$$

What is the asymptotic runtime?

tree

$$W(n) = 2W(n/2) + c_1 n^2 + c_2$$

$$= \sum_{i}^{\lg n} \left(c_1 \frac{n^2}{2^i} + 2^i c_2\right)$$

$$= c_1 n^2 \sum_{i=0}^{\lg n} \frac{1}{2^i} + c_2 \sum_{i=0}^{\lg n} 2^i$$

$$< 2c_1 n^2 + 2c_2 n$$

$$\in O(n^2)$$

So what if branching factor is not 2?

$$W(n) = 4W\left(\frac{n}{2}\right) + O(n)$$

costs

- level 0: $c_1 n + c_2$
- level 1: $4\left(c_1 \frac{n}{2} + c_2\right)$
- level 2: $16\left(c_1 \frac{n}{4} + c_2\right)$
- level i ?

$$4^i \left(c_1 \frac{n}{2^i} + c_2\right)$$

still $\lg n$ levels:, so $W(n)$ is:

$$= c_1 n \sum_{i=0}^{\lg n} \left(\frac{4}{2}\right)^i + c_2 \sum_{i=0}^{\lg n} 4^i$$

$$< 2c_1 n^2 + \frac{4}{3} c_2 4^{\lg n}$$

$$\left(\text{since } \sum_{i=0}^{n} \alpha^i < \frac{\alpha}{\alpha - 1} \cdot \alpha^n\right)$$

$$= 2c_1 n^2 + \frac{4}{3} c_2 2^{\lg n} 2^{\lg n}$$

$$= 2c_1 n^2 + \frac{4}{3} c_2 n^2$$

$$\in O(n^2)$$