CSCI 104L Lecture 1: Introduction and Runtime

An Abstract Data Type (or ADT) explains **what** you want, but not **how** you achieve it. It kind of looks like a header file. It states which functions are used to interact with the data.

- A map ADT:
  1. Add(key,value)
  2. Remove(key)
  3. Lookup(key)
- Here are the key learning goals of this class:
  1. Learn techniques for how to implement data structures which are efficient. A lot of this will draw on the mathematical material from CSCI 170.
  2. Learn mathematical techniques to analyze complicated data structures.
  3. Learn how to identify which data structure will best suit your needs.
  4. Learn to think about code abstractly, separating the "what" from the "how"
  5. Learn to utilize ADTs to specify the functionality of what you want.
  6. Learn good programming practice in Object-Oriented Design.

Runtime Review

- $f(n)$ is $O(g(n))$ means $f(n) \leq c \cdot g(n)$, for all $n \geq n_0$, for some constants c, n_0. That is, our algorithm never takes more time than $g(n)$ times a constant for sufficiently large inputs.
- $f(n)$ is $\Theta(g(n))$ means $f(n)$ is $O(g(n))$ and $f(n)$ is $\Omega(g(n))$. This means that, up to constant factors, f and g are the same function.
- Big-Oh notation measures the growth rate of an algorithm, ignoring constant factors, and focusing on very large inputs. We typically consider the worst-case.
- With a for loop, if iteration i takes $T_i(n)$ time for i from 0 to $n-1$, then the runtime would be $\sum_{i=0}^{n-1} T_i(n)$. Note that each iteration may take a different amount of time.

The following sums will come up in analysis and may prove useful to you.

- $\sum_{i=0}^{n} \theta(i^p) = \Theta(n^{p+1})$. This is a general form of the arithmetic series.
- $\sum_{i=0}^{n} c^i = \frac{c^{n+1}-1}{c-1} = \Theta(c^n)$. This is called the geometic series.
- $\sum_{i=1}^{n} \frac{1}{i} = \Theta(\log n)$. This is called the harmonic series.

Exercise 1. Calculate the runtime:

```
for (int i = 0; i < n; i++)
  if (a[i][0] == 0)
    for (int j = 0; j < i; j++)
      a[i][j] = i*j;
```

Exercise 2. Calculate the runtime:

```
for (int i = 1; i < n; i *= 2)
  for (int j = 0; j < i; j++)
    a[i][j] = i*j;
```

Exercise 3. Calculate the runtime:

```
for (int i = 0; i < n; i++)
  if (i == 0)
    for (int j = 0; j < n; j++)
      a[i][j] = i*j;
```

Exercise 4. What is the running time of the following code?

```
int lo = 0, hi = len-1, mid;
while(lo <= hi) {
    mid = (hi+lo)/2;
    if (b[mid]==t) return mid;
    else if (t < b[mid]) hi = mid-1;
    else lo = mid+1;
}
```

Exercise 5. Calculate the runtime:

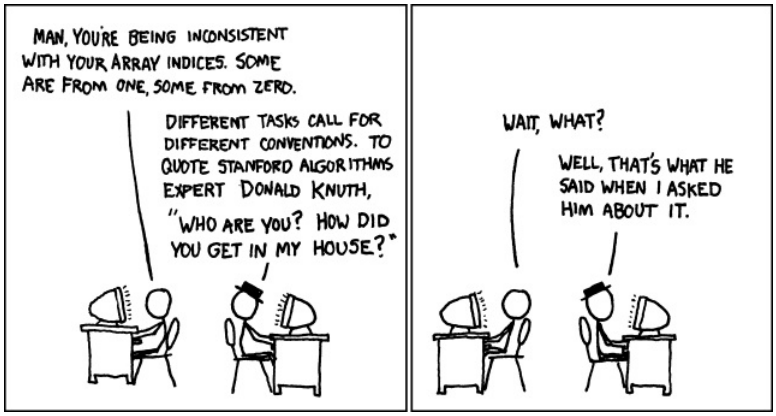
```
for (int i = 0; i < n; i++)
  if ((i % 2) == 0)
    for (int j = 0; j < n; j++)
      a[i][j] = i*j;
  else
    a[i][0] = i;
```

Exercise 6. Calculate the runtime:

```
for (int i = 1; i < n; i++)
  for (int j = 0; j < n; j += i)
    a[i][j] = i*j;
```

Exercise 7. Calculate the runtime:

```
for (int i = 0; i < n; i++)
  for (int j = i; j < n; j++)
    for (int k = i; k < j; k++)
      a[i][j][k] = i*j*k;
```



XKCD # 163: His books were kinda intimidating; rappelling down through his skylight seemed like the best option.