Chapter 1
Analysis Motivation and ADTs
Careful with your analysis.

Algorithm Design and Analysis (Fall 2021)

Christian A. Duncan
School of Engineering
Quinnipiac University

Objectives

1. Explain importance of careful analysis
2. Review Abstract Data Types

Password Generator

Problem

- We want to generate a LONG random password of N characters (A-Z).
- We can use the following pseudocode:

```
Create an initially empty password: result
For i in 1 to N:
   Create a random letter from A-Z
   Add it to result
Return the result
```

- Under Course Materials, you will see three implementations (GenPass.java)

Let us take a quick look at this code.

Password Generator

Problem

- We want to generate a LONG random password of *N* characters (A-Z).
- We can use the following pseudocode:

```
Create an initially empty password: result
For i in 1 to N:
   Create a random letter from A-Z
   Add it to result
Return the result
```

- Under Course Materials, you will see three implementations (GenPass.java)

Breakout Time:

- *Group size: about 4*
- *Time: 5-10 minutes*
- *Ponder: Which technique A, B, or C is most efficient? Or are they all about the same?*

Password Generator

Problem

- We want to generate a LONG random password of N characters (A-Z).
- We can use the following pseudocode:

```
Create an initially empty password: result
For i in 1 to N:
   Create a random letter from A-Z
   Add it to result
Return the result
```

- Under Course Materials, you will see three implementations (GenPass.java)

Share your thoughts

Which method is most efficient?

- A
- B
- C
- All are about the same

Password Generator

Problem

- We want to generate a LONG random password of N characters (A-Z).
- We can use the following pseudocode:

```
Create an initially empty password: result
For i in 1 to N:
   Create a random letter from A-Z
   Add it to result
Return the result
```

- Under Course Materials, you will see three implementations (GenPass.java)

Solution:

- *Method B is far more efficient!!*
- *Why? Let us run all three with a large value of N.*
- *Why? Let us see that in the next slide.*

Password Generator (Explanation)

Method A Code

```
1: String result = "";  // The result
2: for (int i = 0; i < N; i++)
3:     result += (char) ('A' + rand.nextInt(26));
4: return result;
```

- This program goes through the loop N times
  - Each step takes $\Theta(1)$ time.
  - So, the algorithm takes $\Theta(N)$ time.
- Where is the error in that logic?

Password Generator (Explanation)

Method A Code

```
1: String result = "";  // The result
2: for (int i = 0; i < N; i++)
3:     result += (char) ('A' + rand.nextInt(26));
4: return result;
```

- This program goes through the loop N times
  - Each step takes $\Theta(1)$ time.
  - So, the algorithm takes $\Theta(N)$ time.
- Where is the error in that logic?
  Line 3 does NOT take $\Theta(1)$ time.

Password Generator (Explanation)

Method A Code

```
1: String result = "";  // The result
2: for (int i = 0; i < N; i++)
3:     result += (char) ('A' + rand.nextInt(26));
4: return result;
```

- This program goes through the loop N times
  - Each step takes $\Theta(1)$ time.
  - So, the algorithm takes $\Theta(N)$ time.
- Where is the error in that logic?
  Line 3 does NOT take $\Theta(1)$ time.
- The assumption was wrong.
- Intuition is great, but it can be deceptive.
- You do proofs to convince yourself (or someone else) that your idea is correct.

Password Generator (Explanation)

Method A Code

```
1: String result = "";  // The result
2: for (int i = 0; i < N; i++)
3:     result += (char) ('A' + rand.nextInt(26));
4: return result;
```

- This program goes through the loop N times
  - Each step takes $\Theta(1)$ time.
  - So, the algorithm takes $\Theta(N)$ time.
- Where is the error in that logic?
  Line 3 does NOT take $\Theta(1)$ time.
- The assumption was wrong.
- Intuition is great, but it can be deceptive.
- You do proofs to convince yourself (or someone else) that your idea is correct.
- Note, in Method B, the equivalent line does take $\Theta(1)$ time.

Abstract Data Types

Terms

- Type: A collection of values
- Data Type: Types with operations on them (e.g. Strings)
- Abstract Data Type: The specification of a Data Type independent of implementation details
- Data Structure: The implementation of an ADT

Example: Dictionary ADT

- Create Dictionary
- Insert entry (key/value pair)
- Delete entry (key)
- Replace entry (key/value pair)
- Get entry (key) returns value
- Iterate through entrys

Note: The syntax will depend on the language and APIs for the specific implementation.

Example: Dictionary (Java Map, Fruit.java)

```
Map<String,Double> fruit =
   new HashMap<String,Double>();
fruit.put("Apples", 1.29);   // Insertion
fruit.put("Bananas", 0.49);
fruit.put("Pears", 1.45);
fruit.put("Kiwi", 3.45);
Double pears = fruit.get("Pears");  // Searching
fruit.put("Apples", 1.39);   // Replacing
fruit.remove("Kiwi");        // Deletion

// Iteration
for (Map.Entry<String, Double> entry: fruit.entrySet()) {
    System.out.println(entry.getKey() + ": " +
                       entry.getValue());
}
```


Example: Dictionary (Python, dictionary.py)

```
fruit = { "Apples": 1.29, "Bananas": 0.49,
          "Pears": 1.45, "Kiwi":3.45 }
print(fruit["Pears"])       # Searching
fruit["Oranges"] = 1.35     # Insertion
fruit["Apples"] = 1.39      # Ins. w/ Replacement
del fruit["Kiwi"]           # Deletion
for f,p in fruit.items():   # Iteration
    print(f,p)
```