Ad-Hoc Problems

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Ad-Hoc Problems

Your Objectives:

- Be ready to solve classes of problems that involve cultural knowledge,
- be familiar with games that often show up in problems,
- solve simple string manipulation problems,
- know how to tell if a problem should be delayed,
- learn some skills that will be useful the other kinds of problems.

What is an Ad-Hoc Problem, Anyway?

- Ad-Hoc = Latin: “for this”
- Solution involves more problem-specific features than general algorithms.
- Common problem styles:
  - Simulations involving games such as card games, chess, checkers.
  - Other simulations of processes.. “just follow the instructions”
  - Simple string manipulations (anagrams, palindromes, etc.)
  - Problems built to waste time.

Card Games

- Usually Poker Cards
  - Four suits: Hearts (♡), Diamonds (♢), Clubs (♣), Spades (♠).
  - Values: Numbers 2–10, Ace (A), Jack (J), Queen (Q), King (K), maybe Joker
- Poker Hands (usually 5 cards):

Two of a Kind two of any one value

Two Pairs

Three of a Kind

Full House A pair and a three of a kind

Flush Five cards of the same suit

Straight Five cards in consecutive order (e.g., 8,9,10,J,K)

Straight Flush A combination of straight and flush. A Royal Flush has the Ace as well.

Chess and Checkers

- I'm going to assume you know these; ask Google if you don't.
- Problems involving Chess pieces (especially the Knight)
- See Problem C in the 2019 World Final Problem Set for a WF class Checkers problem!

Strings

- Anagrams: using the same letters to write a different word.
  - Greek: $\alpha\nu\alpha$ = "again".
  - Example: "Doctor Who" $\rightarrow$ "Torchwood"
  - Sorting the letters of two strings can detect if they are anagrams.
- Palindromes: reverse the word/sentence and get the same one back.
  - Madam, I'm Adam
  - A man, a plan, a canal: Panama
  - aibohphobia (the fear of palindromes)
  - Note that we often exclude capitals and punctuation from our consideration!

Dates

- A rich mine of edge cases to trip up problem solvers!
- Be sure someone on your team knows Python or Java!
- Be careful with assumptions — periods may cross day / month / year boundaries.
- See `strftime` if you just need to format time values.

Time Wasting

- These problems tend to have long, tedious solutions.
- Code efficiency is often not an issue, but coder efficiency will be.
- Make sure your team has a strategy (e.g., save it for last? do it while the lead programmer takes a break?)

Final Thoughts

- Life skill: Pay Attention to Detail!
- Question your assumptions if you get stuck.
- Challenge: try to never get a compile error.