Technical Interview Workshop

Winter ‘19, CS32 LA Workshop #2

Presented by Royson Lin and Julia Baylon

Part 1: Interviewing - The Basics

Being in CS32...

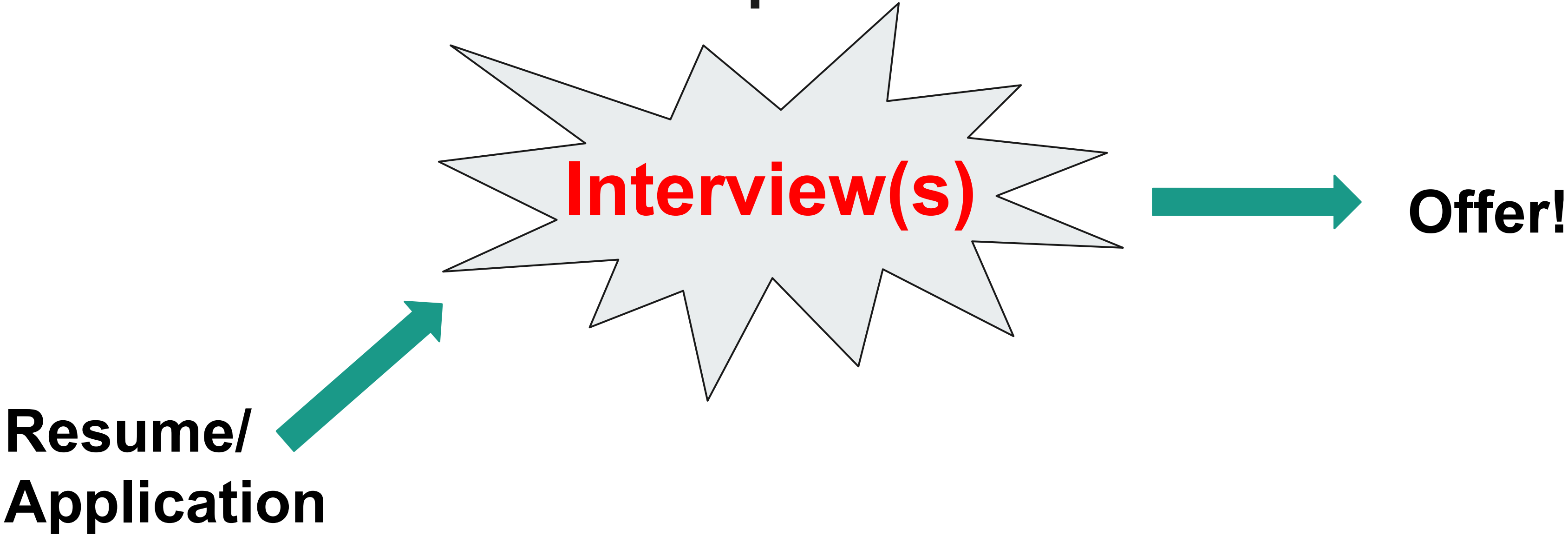
*at job interview
"What are some reasons you'd be a valuable asset to our company?"

Me:

...we understand it might be intimidating.

But it’s a great opportunity to show off your skills!

The Hunt for Internships



Resume/ Application

Interview(s)

Offer!

(Typical) Interview Structure

- 45 minutes total
    - Behavioral questions (5-10 min)
        - Resume
        - Skills
    - Coding (30-35 min)
        - 1-2 questions
    - Questions (5 min)

Things to know/do beforehand

1. **Data structures***
2. **Algorithms and their Big O complexities***
3. How to communicate effectively!
4. Research the company and position you’re applying for
5. How the interviews work (what?)
    a. More on this in the next slide...

*You’ll learn these in CS32 :)

What you may not realize about interviews...

- The interviewers are not out to get you
  - Ask for hints if you need them
  - They want to see how you think, not just whether you can get the right answer
- Be polite!
  - Show them you’re someone they want to work with

Helpful Technical Tips

1. Before starting to code, go through a sample run with the interviewer to make sure you understand the question. **Clarify any ambiguities.**

2. Break down the problem and let the interviewer know how you are approaching it. Write out pseudo-code.

3. As you write, **keep talking out loud** to let the interviewers know what you are doing. This is very important!

    a. They might give feedback - listen to them, they are trying to help!!

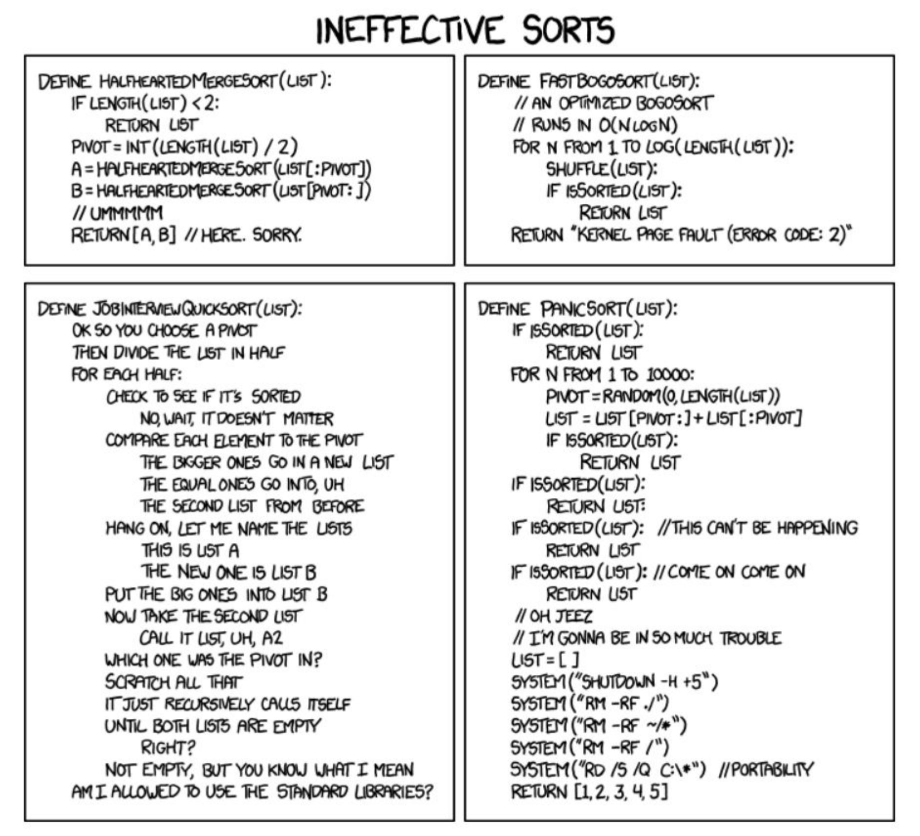
Helpful Technical Tips (continued)

4. TEST TEST TEST. Once you finish, go through a couple test cases to show your solution works. You don’t have to worry too much about edge cases at the beginning, but be aware of them—your interviewer might ask about them!

Resources and order of preference (our rec)

1. Carey's slides
2. *Cracking the Coding Interview*
3. LeetCode, HackerRank, etc.
    a. After getting the necessary background from the previous steps, continue to practice as much as you can here
4. Pramp and interviewing.io—recommended by UPE
    a. Mock interview websites

So what does it look like?



INEFFECTIVE SORTS

```
DEFINE HALFHEARTEDMERGESORT(LIST):
    IF LENGTH(LIST) < 2:
        RETURN LIST
    PIVOT = INT(LENGTH(LIST) / 2)
    A = HALFHEARTEDMERGESORT(LIST[:PIVOT])
    B = HALFHEARTEDMERGESORT(LIST[PIVOT:])
    // UMMMMM
    RETURN [A, B] // HERE. SORRY.
```

```
DEFINE FASTBOGOSORT(LIST):
    // AN OPTIMIZED BOGOSORT
    // RUNS IN O(NLOGN)
    FOR N FROM 1 TO LOG(LENGTH(LIST)):
        SHUFFLE(LIST):
        IF ISSORTED(LIST):
            RETURN LIST
    RETURN "KERNEL PAGE FAULT (ERROR CODE: 2)"
```

```
DEFINE JOBINTERVIEWQUICKSORT(LIST):
    OK SO YOU CHOOSE A PIVOT
    THEN DIVIDE THE LIST IN HALF
    FOR EACH HALF:
        CHECK TO SEE IF IT'S SORTED
            NO, WAIT, IT DOESN'T MATTER
        COMPARE EACH ELEMENT TO THE PIVOT
            THE BIGGER ONES GO IN A NEW LIST
            THE EQUAL ONES GO INTO, UH
            THE SECOND LIST FROM BEFORE
        HANG ON, LET ME NAME THE LISTS
            THIS IS LIST A
            THE NEW ONE IS LIST B
        PUT THE BIG ONES INTO LIST B
        NOW TAKE THE SECOND LIST
            CALL IT LIST, UH, A2
        WHICH ONE WAS THE PIVOT IN?
        SCRATCH ALL THAT
        IT JUST RECURSIVELY CALLS ITSELF
        UNTIL BOTH LISTS ARE EMPTY
            RIGHT?
        NOT EMPTY, BUT YOU KNOW WHAT I MEAN
    AM I ALLOWED TO USE THE STANDARD LIBRARIES?
```

```
DEFINE PANICSORT(LIST):
    IF ISSORTED(LIST):
        RETURN LIST
    FOR N FROM 1 TO 10000:
        PIVOT = RANDOM(0, LENGTH(LIST))
        LIST = LIST[PIVOT:] + LIST[:PIVOT]
        IF ISSORTED(LIST):
            RETURN LIST
    IF ISSORTED(LIST):
        RETURN LIST:
    IF ISSORTED(LIST): //THIS CAN'T BE HAPPENING
        RETURN LIST
    IF ISSORTED(LIST): //COME ON COME ON
        RETURN LIST
    // OH JEEZ
    // I'M GONNA BE IN SO MUCH TROUBLE
    LIST = [ ]
    SYSTEM("SHUTDOWN -H +5")
    SYSTEM("RM -RF ./")
    SYSTEM("RM -RF ~/*")
    SYSTEM("RM -RF /")
    SYSTEM("RD /S /Q C:\*") //PORTABILITY
    RETURN [1, 2, 3, 4, 5]
```

Question #1: Reverse Integer

Given a number, return its reverse.

Function Header:

```
int reverse(int n);
```

Example:

Input: 123, Output: 321

Input: 9, Output: 9

Input: -23, Output: -32

Question #1: Solution (C++)

```
#include <iostream>
using namespace std;

int reverse(int n) {
  int output = 0;
  int sign = 1;
  if (n < 0) {
    sign *= -1;
    n *= -1;
  }
  while(n != 0) {
    output = output*10 + n%10;
    n /= 10;
  }
  return output*sign;
}
```

```
int main() // for testing
{
  cout << reverse(12345) << endl;
}
```

Question #2: TwoSum

Given an array of ints, return whether two of the elements can add up to a given sum.

Function header: (Python bc that’s the only one I remember how to do)

```
def twoSum(a, size, sum):
```

Example:

For a = [1, 3, 5], size = 3, sum = 6 → True

For a = [1, 3, 5], size = 3, sum = 2 → True

Question #2: Solution

```
def twoSum(a, size, sum):
    complements = {}
    for i in range(size):
        complements[a[i]] = sum - a[i]
    for i in range(size):
        if sum - a[i] in complements: # is one of the keys
            return True
    return False
```

```
def main(): # for testing
    a = [1, 3, 5]
    print(twoSum(a, 3, 6)) # True
    print(twoSum(a, 3, 2)) # True
    print(twoSum(a, 3, 1)) # False

if __name__ == "__main__":
    main()
```

Part 2: Code Tracing

What’s it for?

- Helpful for manual debugging of code
- Allows you to quickly figure out what a snippet of code is supposed to do
  - Especially helpful for internships & examining large code bases
- CS32!

Tips

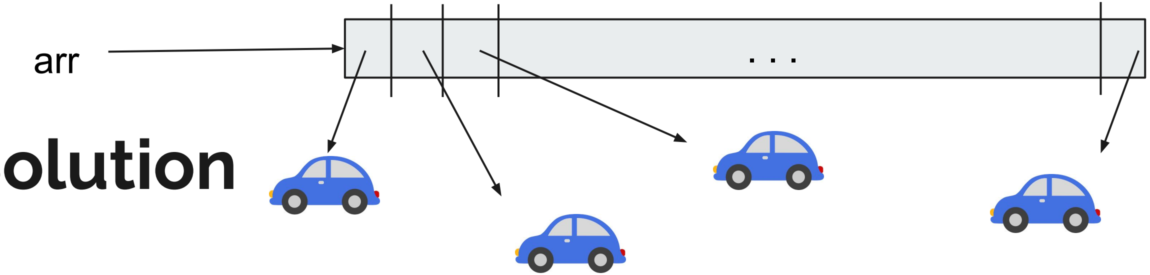
- Run through examples, line by line
- Be especially careful with loops
  - **Draw a table** (or just have **boxes**) to keep track of values of variables
- Follow order of construction/destruction carefully!
- **Draw pictures** for pointers, data structures, and complicated pieces of code!
  - You'll learn all about these data structures in CS32

Question #1: Pointers!

```
#include <iostream>
#include <string>
using namespace std;

class Car {
public:
  Car() {
    m_name = new string;
    *m_name = "anonymous";
    cout << "Hello!" << endl;
  };
  ~Car() {
    // Implement this!
  };
private:
  string * m_name;
};
```

```
int main() {
  int nCars = 15;
  Car ** arr = new Car *[nCars];
  for(int i = 0; i < nCars; i++) {
    arr[i] = new Car();
  }

  // Implement deallocating the array!

  cout << "Done!" << endl;
}
```

Question #1: Solution



```
#include <iostream>
#include <string>
using namespace std;

class Car {
public:
  Car() {
    m_name = new string;
    *m_name = "anonymous";
    cout << "Hello!" << endl;
  };
  ~Car() {
    delete m_name;
  };
private:
  string * m_name;
};
```

```
int main() {
  int nCars = 15;
  Car ** arr = new Car *[nCars];
  for(int i = 0; i < nCars; i++) {
    arr[i] = new Car();
  }

  for (int i = 0; i < nCars; i++) {
    delete arr[i];
  }
  delete[] arr;

  cout << "Done!" << endl;
}
```

Question #2: What’s the output of this program?

```
int main()
        {Winterfell w;}
```

```
class Winterfell {
        public:
                Winterfell () : ned(“Ned”)
                        { cout << “Winter is coming.” << endl;}
                Winterfell (const Winterfell& other)
                        { cout << “The north remembers.” << endl;}
                ~Winterfell ()
                        { cout << “No spoilers here.” << endl;}
        private:
                Stark ned;
};
```

```
class Stark {
        public:
                Stark ()
                        { cout << “...” << endl;}
                Stark (string name)
                        { cout << “My name is “ << name << endl;}
                Stark (const Stark& other)
                        { cout << “Direwolves!!” << endl;}
                ~Stark ()
                        { cout << “:(” << endl;}
};
```

Walkthrough

- Remember the order of construction and destruction!
  - Objects are constructed from the “inside out”
  - Member variables first!

```
class Winterfell {
     public:
          Winterfell () : ned(“Ned”)
               { cout << “Winter is coming.” << endl;}
          Winterfell (const Winterfell& other)
               { cout << “The north remembers.” << endl;}
          ~Winterfell ()
               { cout << “No spoilers here.” << endl;}
     private:
          Stark ned;
};
```

Walkthrough

- Since our Winterfell class has a Stark object, we need to construct the Stark object first.
    - NOTE: if the Winterfell class had a Stark **pointer**, rather than the object itself, this code would not need to run!

```
class Stark {
    public:
        Stark ()
            { cout << “...” << endl;}
        Stark (string name)
            { cout << “My name is “ << name << endl;}
        Stark (const Stark& other)
            { cout << “Direwolves!!” << endl;}
        ~Stark ()
            { cout << “:(” << endl;}
};
```

Walkthrough

- Which constructor would run?

```
class Stark {
    public:
         Stark ()
              { cout << “...” << endl;}
         Stark (string name)
              { cout << “My name is “ << name << endl;}
         Stark (const Stark& other)
              { cout << “Direwolves!!” << endl;}
         ~Stark ()
              { cout << “:(” << endl;}
};
```

Walkthrough

- Go back to the Winterfell class’s constructor.
- Notice we’re constructing ned with the string “Ned”!

```
class Winterfell {
    public:
        Winterfell () : ned(“Ned”)
            { cout << “Winter is coming.” << endl;}
        Winterfell (const Winterfell& other)
            { cout << “The north remembers.” << endl;}
        ~Winterfell ()
            { cout << “No spoilers here.” << endl;}
    private:
        Stark ned;
};
```

Walkthrough

- So now, we know the second constructor would run.

```
Output:
        My name is Ned
```

```
class Stark {
    public:
        Stark ()
            { cout << “...” << endl;}
        Stark (string name)
            { cout << “My name is ” << name << endl;}
        Stark (const Stark& other)
            { cout << “Direwolves!!” << endl;}
        ~Stark ()
            { cout << “:(” << endl;}
};
```

Walkthrough

- Now that the member variable is constructed, the actual class needs to be constructed!

```
Output:
        My name is Ned
        Winter is coming.
```

```
class Winterfell {
    public:
        Winterfell () : ned(“Ned”)
            { cout << “Winter is coming.” << endl;}
        Winterfell (const Winterfell& other)
            { cout << “The north remembers.” << endl;}
        ~Winterfell ()
            { cout << “No spoilers here.” << endl;}
    private:
        Stark ned;
};
```

Are we done yet?

Walkthrough

- Nope. Now we need to destruct the objects we made!

```
Output:
        My name is Ned
        Winter is coming.
```

```
int main()
    {
        Winterfell w;
    }  //destruction time!
```

Walkthrough

- First, the outer class's destructor would run.

```
Output:
        My name is Ned
        Winter is coming.
        No spoilers here.
```

```
class Winterfell {
     public:
          Winterfell () : ned("Ned")
               { cout << "Winter is coming." << endl;}
          Winterfell (const Winterfell& other)
               { cout << "The north remembers." << endl;}
          ~Winterfell ()
               { cout << "No spoilers here." << endl;}
     private:
          Stark ned;
};
```

Walkthrough

- Then, the member variables.

```
Output:
        My name is Ned
        Winter is coming.
        No spoilers here.
        :(
```

```
class Stark {
    public:
        Stark ()
            { cout << “...” << endl;}
        Stark (string name)
            { cout << “My name is “ << name << endl;}
        Stark (const Stark& other)
            { cout << “Direwolves!!” << endl;}
        ~Stark ()
            { cout << “:(” << endl;}
};
```

Code tracing is super useful!

- Some interviews will ask you to trace through code
- And you might have to do code tracing on CS 32 exams to find bugs…. 👀

Good luck!

AM I READY FOR THIS INTERVIEW?
FALSE. THE QUESTION IS, IS THIS INTERVIEW READY FOR ME?
memegenerator.net