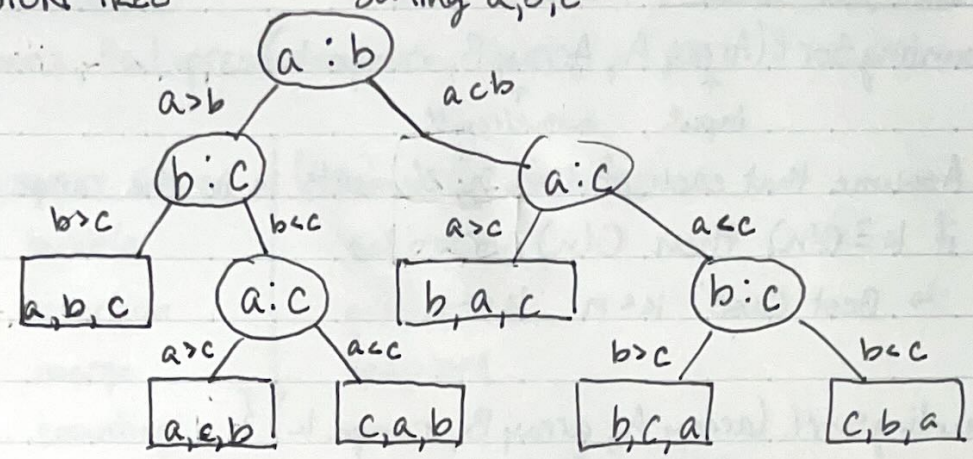
Date 2025 03 05

DECISION TREE

Sorting a,b,c



- minimum of n! leaves
- height $h = 2^h$ leaves
- $h \geq \lg(n!)$
  $h \geq (n \lg n)$
  ↳ This is why comparision-based sorts are lower-bounded at $O(n \log n)$

valgrind ./p1 input
↑ To test for mem leaks

COUNTING SORT (Seward 1954)

CountingSort(Array A, Array B, range k)

↑ input ↑ output/result

- Assume that each of the $\underline{n}$ elements is in the range 0...k
  if $k \in O(n)$ then $O(n)$ sort
  ↳ Best Case: $k < n$ $k << n$

Code:

```
countingsort (array A, array B, range k) {
    newarray C[k+1] = {0};
    for (i=0; i < A.length(); i++) {
        C[A[i]]++;
    }
    for (i=1; i<=k; i++) {
        C[i] += C[i-1];
    }
    for (i = A.length()-1; i >=0; i--) {
        B[C[A[i]]-1] = A[i];
        C[A[i]]--;
    }
}
```

Voting Section

Sum the votes to find indexes

The sort (based on accumulated knowledge)

↑ MTG REF!

Example

A: 0_A 3_A 1_A 2_A 2_B 1_B 0_B 0_C 1_C 2_C 3_B 3_C k=3 0...3

0 1 2 3 4 5 6 7 8 9 10 11

C: 0 0 0 0 => 1 1

B:

0 1 2 3 4 5 6 7 8 9 10 11

3c true

Date

STABLE SORT

- A sort that preserves order incase of ties

STABLE SORTS	(UN)(NON) STABLE
bubble	selection
insertion	quicksort
merge	heapsort
counting	
bucket	

RADIX SORT (Comrie 1929)

- 1 "digit" at a time from lsb to msb

```
radixSort(A, d) {                    d = # of 'digits'
    for (i=1; i=d; i++) {            k = possible range for each digit
        use an "O(n)" stable sort to sort array i++
    }
```

(A: Array; d: Can be char)

BUCKET SORT ISAAC + SINGLETON 1956

```
bucketSort(A, n) {
    for(i=0; i<n; i++) {
        insert A[i] into bucket B[A[i]]
    }
    for(i=0; i<n; i++) {
        sort bucket B[i] with insertion sort
    }
    Combine B[0], B[1], ..., B[n-1] back into array
```

* Won't be using bucket or radix, will use a mix of the two

10 buckets for homework

Date

* Having more buckets is better

RADIX BUCKET SORT ← IMPORTANT

n = A.length()

d = # of "digits"

k = radix of "digits"

```
radixBucketSort(A, n, d, k) {
    new Buckets[k]; // k is 10 for range of digits
    for(i=1; i<=d; i++) { // ith digit from lsd to msd
        for(j=0; j<n; j++) {
            r = ith digit from A[j];
            bucket[r].insertBack(A[j]);
        } // for j
        for(j=0; j<k; j++){
            add items from Bucket[j] into A
            clear Bucket[j];
        } // for i
    }
```

A: Array, n: size

Cpp vec → new Buckets[k]

Py list / Java array list

radix part → r = ith digit from A[j]

int r

insertBack: push_back

Example: SSN ints (digits are digits)
digits d = 9
radix k = 10 (0-9)

O(d(n+k))
Best when d<<n k<<n
then O(n)

Do front to back, correct way is for bottom of bucket to go first

RADIX BUCKET SORT EXAMPLE

212 012 023 231 233 310 313 021 111 101

d = 3 k = 4

3 passes starting on
- units digits
- tens digits
- hundreds digits

BUCKETS