

Algorithms and Data Notes

William Traub

Contents

| | | |
|----------|---|----------|
| 1 | January 9, Introduction | 1 |
| 1.1 | What is CS3000: Algorithms and Data | 1 |
| 1.2 | Egg Drop Contest | 1 |
| 1.3 | The Chocolate Coin Problem | 2 |
| 2 | January 13 | 3 |
| 2.1 | Sorting Algorithms | 3 |
| 2.2 | Describing Algorithms | 3 |
| 2.3 | Divide and conquer | 4 |

1 January 9, Introduction

1.1 What is CS3000: Algorithms and Data

The study of how to solve computational problems.

- How to design **efficient algorithms** - Resources: time, space, parallelism

Why do we care about this?

- Improve your problem solving
 - How to attack new problems
 - Which algorithmic tools apply
 - How to compare different solutions
 - How to know if a solution is the best possible
- etc

1.2 Egg Drop Contest

- You want to test your egg parachute (one egg, ladder with n steps)
- You can try dropping your egg odd different steps to see if it breaks

Linear scan

```
LinearScan(1,n):  
H = 0  
While H <= n and egg intact  
    H = H + 1  
    Drop egg from step H  
  
Return H - 1
```

Worst case number of steps: n

Now with two eggs: start in the middle and if it breaks

```
BalancedScan(1,n):  
H = 0  
While H <= n and first egg intact  
    H = H +  $n^{1/2}$   
    Drop egg from step H  
If first egg intact:  
    Return n  
Else:  
    LinearScan(H -  $n^{1/2}$  + 1, H - 1)
```

Worst case time $2\sqrt{n}$

Suppose you have some $t \in \mathbb{N}$ eggs.

With t eggs you can get a worst case of $tn^{1/t}$

With ∞ eggs, use binary search for worst case $\log_2(n)$

1.3 The Chocolate Coin Problem

You have a sack of n coins, one is chocolate and lighter. Use a scale to see which of the two is heavier

2 January 13

2.1 Sorting Algorithms

Take n elements and return them in ascending order. Selection Sort - Find the minimum, swap it

```
SelectionSort(A[1:n]):  
  for j = 1,...,n-1:  
    min_pos = j  
    for k = j+1,...,n:  
      if (A[k] < A[min_pos]):  
        min_pos = k  
    swap A[min_pos] and A[j]
```

Analysis:

- How to prove correctness?
- Analysis of run time

At any index j , the first $j - 1$ elements are sorted and every element with index $i \geq j$ is larger than element $j - 1$.

- $A[1: j - 1]$ contains the $j - 1$ smallest elements of A in order.
- $A[j:n]$ contains the remaining elements of A .

Proof. Base case $j=1$: This is trivially true as $A[1:0]$ is empty and $A[1:n]$ is the entire array.

Suppose the hypothesis is true.

Induction step $j+1$: Between the start of iteration j and $j+1$, we have found a particular element and swapped □

2.2 Describing Algorithms

- Pseudocode: an easily readable, precise, unambiguous description of an algorithm
 - About clarity not format
 - More like comments than code
 - Often avoids the idiosyncratic details of programming languages.

2.3 Divide and conquer

- Split your problem into smaller subproblems
- Solve the subproblems recursively
- Combine the solutions

Key tools

- Recursion
- proof by induction
- runtime analysis
- Θ notation

Mergesort: