CS 361, Lecture 14

Jared Saia
University of New Mexico

Outline

- Bucket Sort
- Midterm Review

Bucket Sort

- Bucket sort assumes that the input is drawn from a uniform distribution over the range $[0, 1)$
- Basic idea is to divide the interval $[0, 1)$ into n equal size regions, or buckets
- We expect that a small number of elements in A will fall into each bucket
- To get the output, we can sort the numbers in each bucket and just output the sorted buckets in order

Bucket Sort

```
//PRE: A is the array to be sorted, all elements in A[i] are between
$0$ and $1$ inclusive.
//POST: returns a list which is the elements of A in sorted order
BucketSort(A){
B = new List[]
n = length(A)
for (i=1;i<=n;i++){
  insert A[i] at end of list B[floor(n*A[i])];
}
for (i=0;i<=n-1;i++){
  sort list B[i] with insertion sort;
}
return the concatenated list B[0],B[1],...,B[n-1];
}
```

Bucket Sort

- Claim: If the input numbers are distributed uniformly over the range $[0,1)$, then Bucket sort takes expected time $O(n)$
- Let $T(n)$ be the run time of bucket sort on a list of size n
- Let n_i be the random variable givingthe number of elements in bucket $B[i]$
- Then $T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$

Analysis

- We know $T(n) = \Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)$
- Taking expectation of both sides, we have

$$\begin{aligned} E(T(n)) &= E(\Theta(n) + \sum_{i=0}^{n-1} O(n_i^2)) \\ &= \Theta(n) + \sum_{i=0}^{n-1} E(O(n_i^2)) \\ &= \Theta(n) + \sum_{i=0}^{n-1} (O(E(n_i^2))) \end{aligned}$$

- The second step follows by linearity of expectation
- The last step holds since for any constant a and random variable X, $E(aX) = aE(X)$ (see Equation C.21 in the text)

Analysis

- We claim that $E(n_i^2) = 2 - 1/n$
- To prove this, we define indicator random variables: $X_{ij} = 1$ if $A[j]$ falls in bucket i and 0 otherwise (defined for all i, $0 \le i \le n-1$ and j, $1 \le j \le n$)
- Thus, $n_i = \sum_{j=1}^{n} X_{ij}$
- We can now compute $E(n_i^2)$ by expanding the square and regrouping terms

Analysis

$$\begin{aligned} E(n_{i^2}) &= E((\sum_{j=1}^{n} X_{ij})^2) \\ &= E(\sum_{j=1}^{n}\sum_{k=1}^{n} X_{ij}X_{ik}) \\ &= E(\sum_{j=1}^{n} X_{ij}^2 + \sum_{1\le j\le n}\ \sum_{1\le k\le n, k\ne j} X_{ij}X_{ik}) \\ &= \sum_{j=1}^{n} E(X_{ij}^2) + \sum_{1\le j\le n}\ \sum_{1\le k\le n, k\ne j} E(X_{ij}X_{ik})) \end{aligned}$$

Analysis

- We can evaluate the two summations separately. X_{ij} is 1 with probability $1/n$ and 0 otherwise
- Thus $E(X_{ij}^2) = 1 * (1/n) + 0 * (1 - 1/n) = 1/n$
- Where $k \neq j$, the random variables X_{ij} and X_{ik} are independent
- For any two *independent* random variables X and Y, $E(XY) = E(X)E(Y)$ (see C.3 in the book for a proof of this)
- Thus we have that

$$\begin{aligned} E(X_{ij}X_{ik}) &= E(X_{ij})E(X_{ik}) \\ &= (1/n)(1/n) \\ &= (1/n^2) \end{aligned}$$

Analysis

- Substituting these two expected values back into our main equation, we get:

$$\begin{aligned} E(n_i^2) &= \sum_{j=1}^{n} E(X_{ij}^2) + \sum_{1 \leq j \leq n} \sum_{1 \leq k \leq n, k \neq j} E(X_{ij}X_{ik})) \\ &= \sum_{j=1}^{n} (1/n) + \sum_{1 \leq j \leq n} \sum_{1 \leq k \leq n, k \neq j} (1/n^2) \\ &= n(1/n) + (n)(n-1)(1/n^2) \\ &= 1 + (n-1)/n \\ &= 2 - (1/n) \end{aligned}$$

Analysis

- Recall that $E(T(n)) = \Theta(n) + \sum_{i=0}^{n-1}(O(E(n_i^2)))$
- We can now plug in the equation $E(n_i^2) = 2 - (1/n)$ to get

$$\begin{aligned} E(T(n)) &= \Theta(n) + \sum_{i=0}^{n-1} 2 - (1/n) \\ &= \Theta(n) + \Theta(n) \\ &= \Theta(n) \end{aligned}$$

- Thus the entire bucket sort algorithm runs in expected linear time

Midterm

- Midterm: Tuesday, March 23rd in class (the Tuesday after spring break)
- You can bring 2 pages of "cheat sheets" to use during the exam. Otherwise the exam is closed book and closed note
- Note that the web page contains links to prior classes and their midterms. *Many of the questions on my midterm will be similar in flavor to these past midterms (and to exercises in the book)!*

Review Session

- I will have a review session on Monday, March 22nd from 5:30-6:30 in FEC 141 (conference room on first floor of FEC)
- Please try to make it to this review session if at all possible

Midterm

- 5 questions, about 20 points each
- Hard but fair
- There will be some time pressure, so make sure you can e.g. solve recurrences both quickly and correctly.
- I expect a class mean of between 60 :(and 70 :) points

Question 1

Collection of true/false questions and short answer on:

- Asymptotic notation: e.g. I give you a bunch of functions and ask you to give me the simplest possible theta notation for each
- Recurrences: e.g. I ask you to solve a recurrence
- Heaps: e.g. I ask you questions about properties of heaps and priority queues
- Sorting Algorithms: heapsort, quicksort, bucketsort, mergesort, (know resource bounds for these algorithms)
- Probability: Random variables, expectation, linearity of expectation, birthday paradox, analysis of expected runtime of quicksort and bucketsort

Question 2

Solving recurrence relations:

- Like problems on hw 4 and Problem 7-3 (Stooge Sort)
- You'll need to know annihilators, change of variables, handling homogeneous and non-homogeneous parts of recurrences, recursion trees, and the Master Method
- You'll need to know the formulas for sums of convergent and divergent geometric series

Question 3

Asymptotic notation:

- Similar to book problems: 3.1-2, 3.1-5, 3.1-7

Question 4

Recurrence proof using induction (i.e. the substitution method):

- You'll need to give base case, inductive hypothesis and then show the inductive step
- Similar to Exercises 7.2-1, 4.2-1 and 4.2-3

Questions 5

Loop Invariant:

- Will give you an algorithm and ask you to give the loop invariant you would use to show it is correct
- You may also need to give initialization, maintenance and termination for your loop invariant
- Similar to the hw problems and in-class exercises on loop invariants

Asymptotic Notation

- Let's now review asymptotic notation
- I'll review for O notation, make sure you understand the other four types
- $f(n) = O(g(n))$ *if there exists positive constants* c *and* n_0 *such that* $0 \leq f(n) \leq cg(n)$ *for all* $n \geq n_0$
- This means to show that $f(n) = O(g(n))$, *you need to give positive constants* c *and* n_0 for which the above statement is true!

Example 1

- Prove that $2^{n+1} = O(2^n)$
- Goal: Show there exist positive constants c and n_0 such that $2^{n+1} \leq c * 2^n$ for all $n \geq n_0$

$$
\begin{aligned}
2^{n+1} &\leq c * 2^n && (1)\\
2 * 2^n &\leq c * 2^n && (2)\\
2 &\leq c && (3)
\end{aligned}
$$

- Hence for $c = 2$ and $n_0 = 1$, $2^{n+1} \leq c * 2^n$ for all $n \geq n_0$

Example 2

- Prove that $n + \sqrt{n} = O(n)$
- Goal: Show there exist positive constants c and n_0 such that $n + \sqrt{n} \leq c * n$ for all $n \geq n_0$

$$
\begin{aligned}
n + \sqrt{n} &\leq c * n && (4)\\
1 + \frac{1}{\sqrt{n}} &\leq c && (5)\\
& && (6)
\end{aligned}
$$

- Hence if we choose $n_0 = 4$, and $c = 1.5$, then it's true that $n + \sqrt{n} \leq c * n$ for all $n \geq n_0$

Example 3

- Prove that $2^{2n} = O(5^n)$
- Goal: Show there exist positive constants c and n_0 such that $2^{2n} \leq c * 5^n$ for all $n \geq n_0$

$$
\begin{aligned}
2^{2n} &\leq c * 5^n && (7)\\
4^n &\leq c * 5^n && (8)\\
(4/5)^n &\leq c && (9)\\
& && (10)
\end{aligned}
$$

- Hence for $c = 1$ and $n_0 = 1$, $2^{2n} \leq c * 5^n$ for all $n \geq n_0$

A Procedure

Goal: prove that $f(n) = O(g(n))$

1. Write down what this means mathematically
2. Write down the inequality $f(n) \leq c * g(n)$
3. Simplify this inequality so that c is isolated on the right hand side
4. Now find a n_0 and a c such that for all $n \geq n_0$, this simplified inequality is true

In Class Exercise

Show that $n2^n$ is $O(4^n)$

- Q1: What is the exact mathematical statement of what you need to prove?
- Q2: What is the first inequality in the chain of inequalities?
- Q3: What is the simplified inequality where c is isolated?
- Q4: What is a n_0 and c such that the inequality of the last question is always true?

Example: Substitution Method

- Consider the following recurrence:

$$T(n) = 2^{2-n} * T(n-1) * T(n-1)$$

where $T(1) = 2$.

- Show that $T(n) = 2^n$ by induction. Include the following in your proof: 1)the base case(s) 2)the inductive hypothesis and 3)the inductive step.

Example: Substitution Method

- Base Case: $T(1) = 2$ which is in fact 2^1.
- Inductive Hypothesis: For all $j < n$, $T(j) = 2^j$
- Inductive Step: We must show that $T(n) = 2^n$, assuming the inductive hypothesis.

$$\begin{aligned} T(n) &= 2^{2-n} * T(n-1) * T(n-1) \\ T(n) &= 2^{2-n} * 2^{n-1} * 2^{n-1} \\ T(n) &= 2^n \end{aligned}$$

where the inductive hypothesis allows us to make the replacements in the second step.

Todo

- Enjoy Spring!
- Study for Midterm