CSE 202—Day 5

Sorting (continued)

Neil Rhodes

UC San Diego
Overview

• Probabilistic analysis
• Killer adversary for Qicksort
Probability Review

- A *sample space* $S$ is a set of *elementary events*
  - Rolls of a die = \{1, 2, 3, 4, 5, 6\} or flipping two coins = \{HH, HT, TH, TT\}
- A *event* is a subset of $S$.
  - Rolling better than a 4 = \{5, 6\}
  - Flipping at least one head = \{HH, HT, TH\}
- A *probability distribution* $Pr$ on a sample space is a function mapping events to real numbers $\in [0..1]$ such that:
  - $Pr\{S\} = 1$
  - $Pr\{A \cup B\} = Pr\{A\} + Pr\{B\}$ for mutually exclusive events
Probability Review

- If $S$ is finite or countably infinite, then the probability distribution is *discrete*.
- If $S$ is finite and each elementary event $e \in S$ has $Pr\{e\} = 1/|S|$, then we have a *uniform probability distribution*.
  - Probability distribution for rolling dice \{1, 2, 3, 4, 5, 6\} is uniform
  - Probability distribution for scores on the GRE \{400, 410, ..., 1590, 1600\} is not uniform.
Probability Review

- A (discrete) random variable $X$ is a function from the elementary events in a (finite or countably infinite) sample space $S$ to the real numbers. Examples:
  - Flip $k$ coins (sample space has $2^k$ elementary events). $X$ is the total number of heads flipped ($0 \leq X(e) \leq k$)
  - Roll two fair dice. $S = \{(1, 1), (1, 2), ..., (6, 6)\}$. $Y$ is the total shown on the dice
The event $X = x$ is defined to be $\{s \in S : X(s) = x\}$

$X + Y$ is a function. For example, $(X + Y)(e) = X(e) + Y(e)$

An *indicator random variable* $I(A)$ associated with event $A$ is 1 if $A$ occurs and 0 otherwise.
Probability Review

- The *Expectation* $E[X]$ is $\sum_{e \in S} X(e) Pr\{e\}$. That is, the weighted average of $X$.
  - Throw a die. $Y$ is the total shown on the die
    $$E[Y] = 1 \cdot Pr\{1\} + 2 \cdot Pr\{2\} + \ldots + 6 \cdot Pr\{6\} = 3.5$$

- **Linearity of Expectation** theorem: $E[X+Y] = E[X] + E[Y]$ (regardless of whether $X$ and $Y$ are dependent or independent)
  - Throw two dice. $Y + Y$ is the total shown.

- The expected value of an indicator random variable is the probability the associated event occurs.
Analysis of Quicksort

Observations:

- After an element \( q_i \) is chosen as a splitter value and compared to other elements in its range, it is never compared again.
- If two elements are separated into different partitions, they will never be compared to each other.
- If we look at a range of sorted elements \( z_i, \ldots, z_j \), \( z_i \) and \( z_j \) are compared to each other only if \( z_i \) or \( z_j \) is chosen as a splitter before any elements in between \( z_i \) and \( z_j \). (Otherwise, if any element between the two is chosen as a splitter, \( z_i \) and \( z_j \) end up in different partitions.)
Average-case

Let $P_n = \{I_1, I_2, \ldots, I_k\}$ be a set of instances of size $n$ ($P_n$ is the sample space). Assume uniform probability distribution ($\Pr\{I\} = 1/k$).

For $I \in P_n$, let $R_n(I)$ be the running time ($R_n$ is a random variable)

Average-case complexity $T_{avg}(n) = \mathbb{E}[R_n]$.

- Expected value of the random variable $R_n$
- Average of $R_n(I_1), R_n(I_2), \ldots, R_n(I_k)$
Probabilistic complexity

For each instance $I$ in $P_n$, let $T_I(x_1, x_2, \ldots, x_l)$ be the running time on instance $I$ when the random choices are $x_1, x_2, \ldots, x_l$. 

$T_I$ is a random variable.

**Probabilistic complexity** $T_{prob}(n) = \text{Max}_{I \in P_n} E[T_I]$.

- The average runtime of the hardest instance
Randomized Quicksort

Let $Z$ be the set of values in a given sorting instance: $A_1, A_2, ..., A_n$. The values in $Z$ are $z_1, z_2, ..., z_n$ where $z_i$ is the $i$th smallest element. Define $Z_{ij}$ as the set of elements $\{z_i, z_{i+1}, ..., z_{j-1}, z_j\}$.

Define an indicator random variable $X_{ij} = I\{z_i \text{ is compared to } z_j\}$

The total number of comparisons that are performed is: $X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$
Randomized Quicksort

The expected number of comparisons for our input instance is:

\[
E[X] = E\left[ \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j} \right]
\]

\[
= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E[X_{i,j}]
\]

by linearity of expectation

\[
= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr\{z_i \text{ is compared to } z_j\}
\]

by def’n of indicator
Randomized Quicksort

The probability that $z_i$ is compared to $z_j$ is the probability that either $z_i$ or $z_j$ is chosen as a pivot before any element which sorts between them. Given the assumption that each pivot is chosen randomly and independently, the probability an item is chosen first from $\{z_i, \ldots, z_j\}$ is $\frac{1}{j-i+1}$

$$Pr\{z_i \text{ is compared to } z_j\} = \frac{1}{j-i+1} + \frac{1}{j-i+1} = \frac{2}{j-i+1}$$
Randomized Quicksort

The expected number of comparisons for any input instance is:

\[ E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} Pr\{z_i \text{ is compared to } z_j\} \]

\[ = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \]

We’re summing distances:

1 pair at distance \( n - 1 \) \( \frac{1 \cdot 2}{n} \)

2 pairs at distance \( n - 2 \) \( \frac{2 \cdot 2}{(n - 2)} \)

... 

n-1 pairs at distance 1 \( \frac{(n - 1) \cdot 2}{2} \)
Randomized Quicksort

The expected number of comparisons for any input instance is:

\[ E[X] = \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \]

\[ < \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k+1} < \sum_{i=1}^{n-1} 2 \sum_{k=1}^{n} \frac{1}{k} \]

\[ \leq \sum_{i=1}^{n-1} 2(\log n + 1) = \sum_{i=1}^{n-1} O(\log n) \]

\[ = (n - 1)O(\log n) = O(n \log n) \]
Average-case Quicksort

Similar argument: $Pr\{z_i \text{ is compared to } z_j\}$ depends on the probability that a particular item is chosen as a pivot. Since all input instances are equally likely, given a particular array location, all elements are equally likely to be in that position.
Callback adversary

A Killer Adversary for Quicksort
This adversary can be used against any quicksort that provides a callback to do the comparison:

- Unix:
  
  qsort(...,&MyComparisonFunction);

- Perl:
  
  foreach(sort mycomparisonRoutine @words)

- STL (calls operator < on vector elements)
Callback adversary

This adversary can be used against any quicksort that provides a callback (allowing the adversary to dynamically determine the relationships between pairs of elements).

The adversary initially leaves all elements unspecified (value of gas). As time goes on and it is forced to specify relative orderings, it will freeze elements into solids.

Intuition: Make the splitter element compare low against all gaseous elements. Forces the partition to be lopsided.
Callback adversary

Comparing:

- When two frozen elements are compared, their values are used to determine ordering.
- When a solid and a gas are compared, the solid is lower.
- When two gas items are compared, one is frozen (and compares lower).

Determining the splitter:

- *Candidate* is last gas element that survived a compare.
- If two gas elements are compared, and one is the candidate, the candidate is frozen.
 Callback adversary

```c
int cmp(const void *px, const void *py) /* per C standard */
{
    const int x = *(const int*)px;
    const int y = *(const int*)py;

    if(val[x]==gas && val[y]==gas)
        if(x == candidate)
            freeze(x);
        else
            freeze(y);
    if(val[x] == gas)
        candidate = x;
    else if(val[y] == gas)
        candidate = y;
    return val[x] - val[y]; /* only the sign matters */
}
```
Example

Compare

gas | gas | ... | gas
Example

Compare

1 | gas | ... | gas (candidate)
Example

Compare

1 gas ... 2