Lecture 25

We will discuss some Exam 2 questions today. We might do a review Monday, but only if enough good questions are sent by Sunday afternoon.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Weight</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>.NET's VM</td>
<td>5</td>
<td>24</td>
</tr>
<tr>
<td>Instruction set</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>Review</td>
<td>54</td>
<td>64</td>
</tr>
</tbody>
</table>

Question 10

Close vote here...luckily there are many interesting randomized graph and network algorithms...

Question 2

What is the smallest value the 24-bit mantissa register can represent?

000000000000000000000001

2\(^{-24}\)...

What if the clock tick is < 2\(^{-25}\) seconds?

Option 1: 0...01 = 2\(^{-24}\) (error > 2\(^{-25}\) per tick)

Option 2: 0...00 = 0 (error = tick length per tick)

So, error per hour is 1 hour!

Exam 2: Question 2

• In Class 16, we saw that the floating point imprecision in representing 0.1 led to an error of 0.0034 seconds per hour in the Patriot missile time calculations. What clock tick unit would maximize the error accumulated per hour? What is the error? This was the easiest question, but no one got it right!

Is this possible?

• Modern Penium ~ 4GHz
  
  Clock tick = 1/4B s = 1/2\(^{32}\)
  
  2\(^7\) times faster than we need!
Question 4
• Explain two reasons why it is easier to write a garbage collector for Python than it is to write a garbage collector for C?

Mark and Sweep

```
active = all objects on stack
while (!active.isEmpty ())
    newactive = { }
    foreach (Object a in active)
        mark a as reachable (non-garbage)
        foreach (Object o that a points to)
            if o is not marked
                newactive = newactive U { o }
    active = newactive

```

Question 9
• Consider modifying the x86 calling convention to put the return value on the top of the stack when a routine returns instead of using EAX to hold the return value. What are the advantages and disadvantages of this change?

Where could it go?

C calling convention: state before ret

Modified calling convention: state before ret

Yikes! ret expects [esp] to be return address

Implications:
Caller: must reserve enough space on stack to hold results
Is this a good idea?

- **Advantages:**
  - Frees up EAX for other things
  - Allows longer return values
  - Multiple results (Python)
  - Return arrays, structures

- **Disadvantages:**
  - Stack access can be a lot slower than registers
  - If caller uses result, it probably needs to copy it into a register anyway

Sample Programs

```python
... x = f(a); ... x = f (g (h (a)))
y = g(b); ... y = g(h(a))
z = f(x + y);
```

Which code fragment could be faster with the new convention?

Why use randomness?

- Avoid worst-case behavior: randomness can (probabilistically) guarantee average case behavior
- Efficient approximate solutions to intractable problems

Randomized Algorithms

Randomized Algorithm

```
Input    Deterministic Computer
         Random bits
Output

www.lavarnd.org (doesn't use lava lamps anymore)
```

Types of Algorithms

- **Monte Carlo**
  - Running time bounded by input size, but answer may be wrong
  - Decision problems: If there is no solution, always returns “no”. If there is a solution, finds it with some probability \( \geq \frac{1}{2} \).
  - Value problems: run for a bounded number of steps, produce an answer that is correct approximation with a bounded probability (function of number of steps)
Types of Random Algorithms

- Las Vegas
  - Guaranteed to produce correct answer, but running time is probabilistic
- Atlantic City
  - Running time bounded by input
  - Can return either “yes” or “no” regardless of correct answer. Correct with probability $\geq 2/3$.

How could this be useful?

Determining $\pi$

Square = 1
Circle = $\pi/4$

The probability a random point in square is in circle:

$\pi/4$

$\pi = 4 \times \text{points in circle/points}$

Find $\pi$

```python
def findPi(points):
    incircle = 0
    for i in range(points):
        x = random.random()
        y = random.random()
        if (square(x - 0.5) + square(y - 0.5) < 0.25):  # 0.25 = $r^2$
            incircle = incircle + 1

    return 4.0 * incircle / points
```

Monte Carlo or Las Vegas?

Results

<table>
<thead>
<tr>
<th>n</th>
<th>4.0</th>
<th>4.0</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2.0</td>
<td>4.0</td>
<td>4.0</td>
</tr>
<tr>
<td>4</td>
<td>3.0</td>
<td>4.0</td>
<td>3.0</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>3.0625</td>
<td>3.125</td>
<td>3.0625</td>
</tr>
</tbody>
</table>

Minimum Cut Problem

- Input: an undirected, connected multigraph $G = (V,E)$
- Output: A cut $(V_1, V_2)$ where $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$ such that number of edges between $V_1$ and $V_2$ is the fewest possible.

Why might this be useful?
Equivalent: fewest edges that can be removed to disconnect $G$. 
Minimum Cut

Size of the min cut must be no larger than the smallest node degree in graph

Randomized Algorithm

• While |V| > 2:
  – Pick a random edge \((x, y)\) from \(E\)
  – Contract the edge:
    • Keep multi-edges, remove self-loops
    • Combine nodes
  • The two remaining nodes represent reasonable choices for the minimum cut sets

Analysis

• Suppose \(C\) is a minimum cut (set of edges that disconnects \(G\))
• When we contract edge \(e\):
  – Unlikely that \(e \in C\)
  – So, \(C\) is likely to be preserved

What is the probability a randomly chosen edge is in \(C\)?

Random Edge in \(C\)?

• \(|C|\) must be \(\leq\) degree of every node in \(G\)
• How many edges in \(G\):
  \(|E| = \text{sum of all node degrees} / 2 \geq n |C| / 2\)

Probability a random edge is in \(C\) \(\leq 2/n\)
Iteration

- How many iterations? $n - 2$
- Probability for first iteration:
  $\Pr(e_1 \notin C) \geq 1 - \frac{2}{n}$
- Probability for second iteration:
  $\Pr(e_2 \notin C | e_1 \notin C) \geq 1 - \frac{2}{(n-1)}$
  ...
- Probability for last iteration:
  $\Pr(e_{n-2} \notin C) \geq 1 - \frac{2}{(n-(n-2)-1)} \geq 1 - \frac{2}{3}$

Probability of finding $C$?

$\geq (1 - \frac{2}{n}) \times (1 - \frac{2}{(n-1)}) \times (1 - \frac{2}{(n-2)}) \times \ldots \times (1 - \frac{2}{3})$

$= \frac{n - 2}{n} \times \frac{n - 3}{(n-1)} \times \frac{n - 4}{(n-2)} \times \ldots \times (\frac{2}{3}) \times (\frac{1}{3})$

Probability of not finding $C$

$= 1 - \frac{2}{(n^2(n-1))}$

Is this good enough?

Probability of not finding $C$ on one trial:

$\leq 1 - \frac{2}{(n^2(n-1))} \leq 1 - \frac{2}{n^2}$

Probability of not finding $C$ on $k$ trials:

$\leq (1 - \frac{2}{n^2})^k$

If $k = cn^2$,

$\Pr$ failure $\leq (\frac{1}{e})^c$

Recall: $\lim_{x \to \infty} (1 - 1/x)^x = \frac{1}{e}$

Charge

- Monday is last class: it will be mostly review if enough good review questions are sent in
- No section or Small Hall hours next week