CS216: Program and Data Representation
University of Virginia Computer Science
Spring 2006 — David Evans

Lecture 25

http://www.cs.virginia.edu/cs216

Question 10

In...		
Pa...		
G...		
Ra...		
.NET's VM	**5**	**24**
Instruction set	**2**	**18**
Review	**54**	**64**

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.

Question 10

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

Impro...		
Partner Assignment Algorithms	**1**	**14**
Graph and Network Algorithms	**12**	**37**
Randomized Algorithms	**9**	**38**
.NET's VM	**5**	**24**
Instruction set	**2**	**18**
Review	**54**	**64**

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!

Question 2

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

000000000000000000000001
2^{-1}… 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!

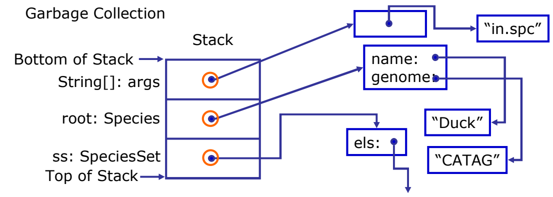
Is this possible?

- 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?

Garbage Collection




```
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
```

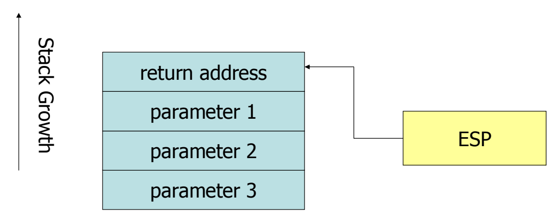
Mark and Sweep

Class 12

Question 9

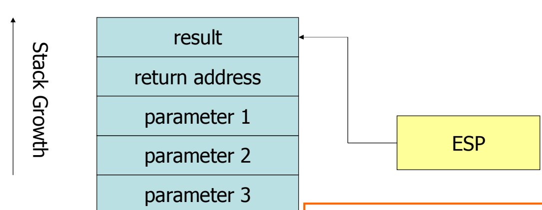
- Consider modifying the x86 calling convention to put the return value on the top of the stack when a routine returns instead of using EAX to hold the return value. What are the advantages and disadvantages of this change?

Where could it go?



C calling convention: state before **ret**

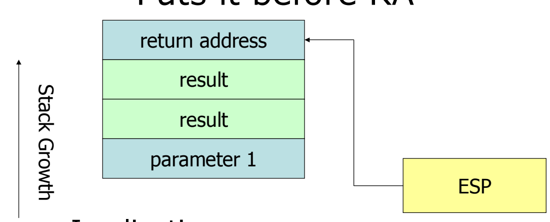
Where could it go?



Modified calling convention: state before **ret**

Yikes! ret expects [esp] to be return address

Puts it before RA



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

```
…
x = f(a);
y = g(b);
z = f(x + y);
```

```
…
x = f (g (h (a)))
…
```

Which code fragment could be faster with the new convention?

Randomized Algorithms

Why use randomness?

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

Randomized Algorithm

Input → Deterministic Computer → Output

↑ Random bits

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 >= ½.
  - 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

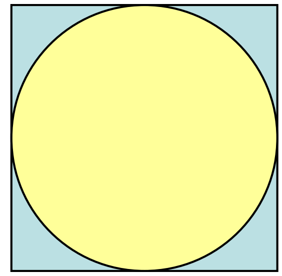
- Las Vegas
  - Guaranteed to produce correct answer, but running time is probabilistic
- Atlantic City
  - Running time bounded by input
  - Can return either "yes" or "no" regardless of correct answer. Correct with probability >= 2/3.

How could this be useful?

Determining π

Square = 1
Circle = $\pi/4$

The probability a random point in square is in circle:
= $\pi/4$

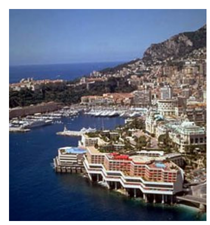


π = 4 * points in circle/points

Find π

```
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

n: 1	4.0	4.0	0.0
n: 2	2.0	4.0	4.0
n: 4	3.0	4.0	3.0
...			
n: 64	3.0625	3.125	3.0625
...			
n:			
...			
n: 16384	3.12622070312	3.14038085938	3.1279296875
n: 131072	3.13494873047	3.14785766602	3.13766479492
n: 1048576	3.14015579224	3.14387893677	3.14112472534

If we wait long enough will it produce an arbitrarily accurate value?

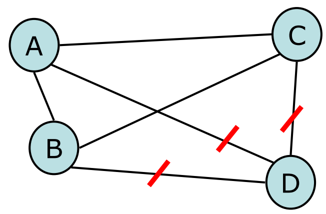
Minimum Cut Problem

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

Why might this be useful?

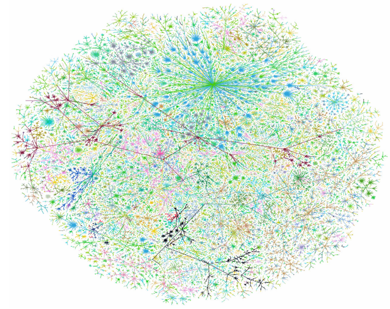
Equivalent: fewest edges that can be removed to disconnect G.

Minimum Cut



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

Internet Minimum Cut



June 1999 Internet graph, Bill Cheswick
http://research.lumeta.com/ches/map/gallery/index.html

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 choosen edge is in C?

Analysis

- Is the final result a cut of G?
- What is the probability we find a minimum cut?

Random Edge in C?

- $|C|$ must be $\leq$ degree of every node in G
- How many edges in G:

$|E|$ = 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:
  $\text{Prob}(e_1 \notin C) \geq 1 - 2/n$
- Probability for second iteration:
  $\text{Prob}(e_2 \notin C \mid e_1 \notin C) \geq 1 - 2/(n\text{-}1)$
- ...
- Probability for last iteration:
  $\text{Prob}(e_{n-2} \notin C) \geq 1 - 2/(n\text{-}(n\text{-}2\text{-}1)) \geq 1 - 2/3$

Probability of finding C?

$\geq (1 - 2/n) * (1 - 2/(n - 1)) * (1 - 2/(n - 2)) \ldots$
$\quad * (1 - 2/3)$
$= (\mathbf{n - 2} / n) * (n - 3/(n - 1)) * (n - 4/(\mathbf{n - 2}))$
$\quad * \ldots * (2/4) * (1/3)$
$= 2 / (n * (n - 1))$

Probability of not finding C
$= 1 - 2/(n*(n\text{-}1))$

Is this good enough?

Probability of not finding C on one trial:
$\leq 1 - 2/(n*(n\text{-}1)) \leq 1 - 2/n^2$

Probability of not finding C on k trials:
$\leq [1 - 2/n^2]^k$
If $k = cn^2$,
Prob failure $\leq (1/e)^c$
Recall: $\lim_{x \to \infty} (1 - 1/x)^x = 1/e$

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