Infinite Loops

- Which of these subroutines terminate for all initial values of n?

```python
def ex1(n):
    while n < 14:
        print n
        n = n + 1

def ex2(n):
    while True:
        print n
        n = n - 1

def ex3(n):
    while n < 21:
        print n
        n = n - 1

def ex4(n):
    while n > 100:
        print n
        n = n + 1

def ex5(n):
    while n > 0:
        print n
        n = n - 1

def ex6(n):
    while False:
        print n
        n = n + 1
```
• What sorts of program would *purposely* have an infinite loop?

• Think about a software-controlled thermostat. It might have a program that looks something like:

```python
def thermostat(low, high):
    while True:
        t = currentTemp()
        if t >= high:
            runAC(4)
        elif t < low:
            runHeat(2)
    thermostat(68, 75)
```

---

**Loop Forever**

• operating systems
• user interfaces
• video games
• process controllers
• robots
Robot Basics

- From a software standpoint, modern robots are just computers.
- Typically, they have less memory and processing power than a standard computer.
- Sensors and effectors under software control.

Standard Robots

- Industrial manufacturing robots.
- Research /hobby robots.
- Demonstration robots.
- Home robots.
- Planetary rovers.
- Movie robots.
Manufacturing

- Often arms, little else.
- Part sorting.
- Painting.
- Repeatable actions.

Research / Hobby

- Pioneer
- Handy Board / Lego
- Segbot
- Stanley
Space Exploration

• Sojourner
• Deep Space Agent

Home Robots

• Roomba.
• Mowers.
• Moppers.
• Big in Japan.
• Nursebots.
• Emergency rescue bots, Aibo.
Demonstration Robots

- Honda: Asimo.
- Toyota: lip robot.
- Sony: QRio.

Sensors and Effectors

- **Sensors:**
  - bump
  - infrared
  - vision
  - light
  - sonar
  - sound

- **Effectors:**
  - motors
  - lights
  - sounds
  - graphical display
  - laser
Simple Learning

- Words: “hello”, “don’t do that”, “sit”, “stand up”, “lie down”, “shake paw”

Example Code

```python
act[0] = 0
act[1] = 0
actions = ["lay6", "sit2", "sit4", "stand2", "stand9"]
lastact = 0
while True:
    cmd = Voice()
    if cmd == "sit":
        doAction(actions[act[0]])
    elif cmd == "stand":
        doAction(actions[act[1]])
    elif cmd == "good Aibo":
        doAction("happy")
    elif cmd == "bad dog":
        doAction("sad sound")
    act[lastact] = (act[lastact] + 1) % 4
```

Trainer: In Words

- For each recognized voice command, there is an associated action program.
- When a voice command is recognized, the corresponding action is taken.
- On “Good Aibo”, nothing needs to change.
- On “Don’t do that”, the most recent command needs a different action program. It is incremented to the next on the list.

Impressive Accomplishment

Honda’s Asimo

- development began in 1999, building on 13 years of engineering experience.
- claimed “most advanced humanoid robot ever created”
- walks 1mph
And Yet...

Asimo is programmed/controlled by people:

- structure of the walk programmed in
- reactions to perturbations programmed in
- directed by technicians and puppeteers during the performance
- no camera-control loop
- static stability

Compare To Kids

Molly

- development began in 1999
- “just an average kid”
- walks 2.5mph even on unfamiliar terrain
- very hard to control
- dynamically stable (sometimes)
Crawl Before Walk

Impressive accomplishment:

- Fastest reported walk/crawl on an Aibo
- Gait pattern optimized automatically

Human “Crawling”

Crawling Behaviors

Perhaps our programming isn’t for crawling at all, but for the desire for movement!
My Research

How can we create smarter machines?

- Programming
  - tell them exactly what to do
  - “give a man a fish...”
- Programming by Demonstration (supervised learning)
  - show them exactly what do do
  - “teach a man to fish...”
- Programming by Motivation (reinforcement learning)
  - tell them what to want to do
  - “give a man a taste for fish...”

Find The Ball Task

Learn:

- which way to turn
- to minimize time
- to see goal (ball)
- from camera input
- given experience.
In Other Words...

• It “wants” to see the pink ball.
  • Utility values from seeing the ball and the cost of movement come from the reward function.
• It gathers experience about how its behavior changes the state of the world.
  • We call this knowledge its transition model.
• It selects actions that it predicts will result in maximum reward (seeing the ball soon).
  • This computation is often called planning.

Exploration/Exploitation

Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

• In RL: A system cannot make two decisions and be one decision maker---it can only observe the effects of the actions it actually chooses to make. Do the right thing and learn.
**Pangloss Assumption**

We are in the best of all possible worlds.

Confidence intervals are on model parameters.

Find the *model* that gives maximum reward subject to the constraint that all parameters lie within their confidence intervals.

Choose actions that are best for this model.

In bandit case, this works out to precisely IE.

Very general, but can be intractable.

*Solvable for MDPs.* (Strehl & Littman 04, 05)

---

**Exploration Speeds Learning**

*Task:* Exit room using bird’s-eye state representation.

*Details:* Discretized 15x15 grid x 18 orientation (4050 states); 6 actions. Rewards via RMAX (*Brafman & Tennenholtz* 02).
Shaping Rewards

• “Real” task: Escape.
• One definition of reward function:
  • -1 for each step, +100 for escape.
  • Learning is too slow.
  • If survival depends on escape, would not survive.
• Alternative:
  • Additional +10 for pushing any button.
  • We call these “Shaping rewards”.

Robotic Example

• State space: image location of button, size of button (3 dimensions), and color of button (discrete: red / green).
• Actions: Turn L / R, go forward, thrust head.
• Reward: Exit the box.
• Shaping reward: Pushing button.
**Pros and Cons of Shaping**

- Can be really helpful.
- Not really the main task, but serve to encourage learning of pertinent parts of the model.
  
  *Example:* Babies like standing up.
- Somewhat risky.
  
  Can “distract” the learner so it spends all its time gathering easy-to-find, but task-irrelevant rewards.
- Learner can’t tell a “real” reward from a shaping reward.

**RL: Sum Up**

- We’re building artificial decision makers as follows:
  
  We define perceptions, actions, and rewards (including shaping rewards to aid learning).
  
  Learner explores its environments to discover:
  
  - What actions do
  - Which situations lead to reward
  - Learner uses this knowledge via “planning” to make decisions that lead to maximum reward.