Recursive functions are equivalent to iterative functions

Just use a stack!

Mihir Mehta

Department of Computer Science
University of Texas, Austin
mihir@cs.utexas.edu

16 October 2014

```
unsigned power (unsigned x, unsigned n) {
  if (n == 0)
    return 1;
  else
    return x * power(x, n - 1);
}

unsigned power (unsigned x, unsigned n) {
  unsigned ret = 1;
  while (n != 0) {
    n = n - 1;
    ret = x * ret;
  }
  return ret;
}
```

```
unsigned power (unsigned x, unsigned n) {
  if (n == 0)
    return 1;
  else {
    unsigned temp = power(x, n / 2);
    if (n % 2 == 1)
      return x * temp * temp;
    else
      return temp * temp;
  }
}
```

```
unsigned power (unsigned x, unsigned n) {
  unsigned ret = 1;
  std::stack<unsigned> s;
  while (n != 0) {
    s.push(n);
    n = n / 2;
  }
  while (!s.empty()) {
    if (s.top() % 2 == 1)
      ret = x * ret * ret;
    else
      ret = ret * ret;
    s.pop();
  }
  return ret;
}
```

```
struct vertex {
  std::vector<vertex *> children;
};

unsigned size(vertex *v) {
  unsigned ret = 1;
  for (unsigned i1 = 0;
       i1 < v->children.size();
       ++i1)
    ret += size(v->children[i1]);
  return ret;
}

struct se {
  vertex *v;
  unsigned seen = 0;
  unsigned size = 1;
};
```

```
unsigned size(vertex *v) {
  unsigned ret = 0; std::stack<se> s;
  se temp; temp.v = v;
  s.push(temp);
  while (!s.empty()) {
    if (s.top().seen <
        s.top().v->children.size()) {
      s.top().size += ret;
      se temp; temp.v = v->children[seen];
      s.push(temp);
    } else {
      ret = s.top().size;
      s.pop();
    }
  }
  return ret;
}
```