Linear Structures

The Problem
Two Sequential Implementations
    Arrays
    Linked Lists

The Problem

Implement this algorithm for random sorted sets

```
initialize set S to empty
size = 0
while size < m do
    t = bigrand() % maxval
    if t is not in S
        insert t into S
        size++
print the elements of S in sorted order
```

A C++ Approach

A Set Implementation

```
class IntSetImp {
public:
    IntSetImp(int maxelems, int maxval);
    void insert(int t);
    int size();
    void report(int *v);
};
```

Algorithm Implementation

```
void gensets(int m, int maxval)
{   int *v = new int[m];
    IntSetImp S(m, maxval);
    while (S.size() < m)
        S.insert(bigrand() % maxval);
    S.report(v);
    for (int i = 0; i < m; i++)
        cout << v[i] << "\n";
}
```

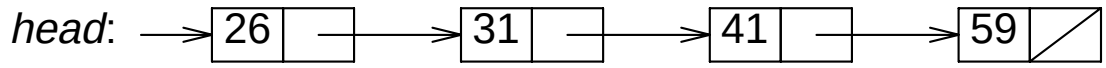
An STL Implementation

```
class IntSetSTL {
private:
    set<int> S;
public:
    IntSetSTL(int maxelems, int maxval) { }
    int size() { return S.size(); }
    void insert(int t) { S.insert(t); }
    void report(int *v)
    {   int j = 0;
        set<int>::iterator i;
        for (i=S.begin(); i!=S.end(); ++i)
            v[j++] = *i;
    }
};
```

(Sorted) Arrays

```
class IntSetArray {
private:
    int n, *x;
public:
    IntSetArray(int maxelms, int maxval)
    {   x = new int[1 + maxelms];
        n = 0;
        x[0] = maxval;
    }
    int size() { return n; }
    void insert(int t)
    {   for (int i = 0; x[i] < t; i++)
            ;
        if (x[i] == t)
            return;
        for (int j = n; j >= i; j--)
            x[j+1] = x[j];
        x[i] = t;
        n++;
    }
    void report(int *v)
    {   for (int i = 0; i < n; i++)
            v[i] = x[i];
    }
};
```

(Sorted) Linked Lists




```
class IntSetList {
private:
    int n;
    struct node {
        int val;
        node *next;
        node(int v, node *p)
            { val = v; next = p; }
    };
    node *head, *sentinel;
    node *rinsert(node *p, int t)
    {   if (p->val < t) {
            p->next = rinsert(p->next, t);
        } else if (p->val > t) {
            p = new node(t, p);
            n++;
        }
        return p;
    }
```

Lists, Cont.

```
public:
  IntSetList(int maxelms, int maxval)
  {   sentinel = head =
                  new node(maxval, 0);
        n = 0;
  }
  int size() { return n; }
  void insert(int t)
      { head = rinsert(head, t); }
  void report(int *v)
  {   int j = 0;
      node *p;
      for (p=head; p!=sentinel; p=p->next)
          v[j++] = p->val;
  }
};
```

Run Times

Experiments ($n = 10^6$)

Structure	Set Size (m)		
	10,000	20,000	40,000
Arrays	0.6	2.6	11.1
Simple Lists	5.7	31.2	170.0
Lists (Remove Recursion)	1.8	12.6	73.8
Lists (Group Allocation)	1.2	5.7	25.4

Advanced Structures

$n = 10^8$, raise m until thrashing.

Structure	Set Size (m)					
	1,000,000		5,000,000		10,000,000	
	Secs	Mbytes	Secs	Mbytes	Secs	Mbytes
STL	9.38	72				
BST	7.30	56				
BST*	3.71	16	25.26	80		
Bins	2.36	60				
Bins*	1.02	16	5.55	80		
BitVec	3.72	16	5.70	32	8.36	52