Quicksort

- Very good runtime performance
  - O(n log n) average, but O(n^2) with bad choices
- Popular C interface

```
void qsort(void *base, size_t nmemb, size_t size,
                           int(*compar)(const void *, const void *));
```

- Easy story
  - Pick a pivot
  - Divide the group into those before and after the pivot
  - Sort the two groups *recursively*
- Lots of good web sites and on-line animations
- Example – sorting linked list of Problem Set 7

```
#include <stdlib.h>

typedef struct node {
    int num;
    struct node *next;
} Node;

/* Compare two nodes */
static int cmpNode(const void *N1, const void *N2) {
  Node *p1 = *((Node **)N1) ;
  Node *p2 = *((Node **)N2) ;
  return p1->next - p2->next ;
}

Node * sortNumberList(Node *N) {
  int size ;      /* Length of list */
  Node **t ;      /* Array to point to elements of list
  int i ;
  Node *p ;

  if (N = (Node *)NULL) {
    return (Node *)NULL ;
  }

  /* Determine the size of the linked list */
  size = 0 ;
  p = N ;
  while (p->next != (Node *)NULL) {
    p = p -> next ;
    ++size ;
  }

  /* Allocate and populate an array to hold pointers to nodes */
  t = (Node **)malloc(size * sizeof(Node *)) ;
  for(i=0, p=N; p->next != (Node *)NULL; ++i, p=p->next) {
    t[i] = p ;
  }

  /* Sort the array */
  qsort((void *)t, size, sizeof(Node *), cmpNode) ;

  /* Rethread the array */
  for(i=0; i<size-1; ++i) {
    t[i]->next = t[i+1] ;
  }
  t[size-1]->next = (Node *)NULL ;

  p = t[0] ;
  free((void *) t) ;
  return p ;
}
```

Search – Finding the one
- Sequential search – O(*n*)
  - But additions are O(1)
- Binary search – O(log *n*)
  - But additions are O(*n*)
- Hashing – O(1)
  - Map keys into "random" integers from 0..M-1, where M is size of hash table
    - Hash function – H(key) → N
  - Collisions will occur
    - Open hashing – Handle collisions within a table
    - Closed hashing – Handle collisions with linked lists
  - Problems with hashing
    - Works best with sets of known sizes
      - Additions can require O(*n*) but amortized cost is O(1)
    - Difficult to list sorted data
  - Applications of hashing
    - Computer memory caches
      - Pentium 4 L1 cache – 8K, 4-way, with 64 byte lines
    - Cryptographic hash
      - SHA hashes of the NSA
- Hash functions – djb2 (Daniel J. Bernstein)

```
unsigned long hash(unsigned char *str) {
    unsigned long hash = 5381 ;
    unsigned char c ;
    while (c = *str++)
      hash = ((hash << 5) + hash) + c
    return hash;
}
```

- Skip lists
  - O(log n) search and O(log n) addition time
  - http://iamwww.unibe.ch/~wenger/DA/SkipList/