Algorithms for Cards

♠ ♣ ♦ ♥

A Deck of Cards

• A deck consists of 52 cards
• A card has a
  – Suit: spades, clubs, hearts, or diamonds
  – Value: 2, 3, 4, ..., 10, Jack, Queen, King, or Ace
• The algorithms we discuss can be made to work with any number of cards

Algorithms Come First

• We’ll look at some common card manipulations and then simulate them
• This will help us design classes for representing cards
• Rule of thumb: first work out your algorithms by hand, and then use your understanding of them to design good programs

Some Card Concerns

• We’ll consider three things
  – How can we sort cards in order from smallest to largest (or from largest to smallest)?
  – How can you cut a deck of cards?
  – How can you shuffle a deck of cards?
• Each of these questions lead to interesting computer science questions

Sorting Your Hand

• Deal 7 cards
  – Ignore suits (for now!)
  – Aces are high
• Put them into sorted order, lowest to highest, left to right
• Try it! How do you do it?

• 4♠ T♣ Q♣ 4♥ 9♦ J♠ A♦

<table>
<thead>
<tr>
<th>4♠</th>
<th>4♥</th>
<th>9♦</th>
<th>T♣</th>
<th>J♠</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q♣</td>
<td>C♣</td>
<td>A♦</td>
<td>4♠</td>
<td>9♦</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>4♥</th>
<th>4♠</th>
<th>9♦</th>
<th>T♣</th>
<th>J♠</th>
</tr>
</thead>
<tbody>
<tr>
<td>9♦</td>
<td>T♣</td>
<td>J♠</td>
<td>4♥</td>
<td>4♠</td>
</tr>
</tbody>
</table>

Sorting: Method 1

• Put the smallest card in your hand face-up on the table
• Repeat this until all your cards are on the table
• Pick up your cards, they’re now sorted!

• 4♠ T♣ Q♣ 4♥ 9♦ J♠ A♦

<table>
<thead>
<tr>
<th>4♠</th>
<th>4♥</th>
</tr>
</thead>
<tbody>
<tr>
<td>T♣</td>
<td>Q♣</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4♠</th>
<th>4♥</th>
</tr>
</thead>
<tbody>
<tr>
<td>T♣</td>
<td>J♠</td>
</tr>
</tbody>
</table>

or

<table>
<thead>
<tr>
<th>4♠</th>
<th>4♥</th>
</tr>
</thead>
<tbody>
<tr>
<td>T♣</td>
<td>Q♣</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4♠</th>
<th>4♥</th>
</tr>
</thead>
<tbody>
<tr>
<td>T♣</td>
<td>J♠</td>
</tr>
</tbody>
</table>
Sorting: Method 2

- Divide your hand into a sorted and unsorted part
- Put the smallest card from the unsorted part onto the end of the sorted part
- Repeat until all your cards are sorted

Two Sorting Methods

- Method 1 is called selection sort
- Method 2 is called insertion sort
- Both of these algorithms are good for sorting small numbers of objects
  - If you’re sorting thousands or millions of things, these methods are too slow
  - There are faster ways to sort: quicksort, mergesort, heapsort, etc.

Selection Sort in Detail

- Use two piles, one called sorted, the other called unsorted
- Initially, everything is in unsorted
- Pick the smallest card from unsorted, and put it face up on top of sorted
- Repeat the previous step until unsorted is empty --- then everything is in sorted order in sorted!

Sorting with Suits

- When two cards have the same value, we can order them if we specify an order for the suits
- There are 24 different ways to order the suits
- Pick one, such as ♠ < ♥ < ♦ < ♣ like
  - This means: 7♠ < 7♥ < 7♦ < 7♣

Cutting Cards

- The standard cut
  - Put the deck on the table
  - Pick up the top half of the deck and place it to the right
  - Pick up the bottom half and put it on top of the other half
- This is easy to do by hand, but tricky to do efficiently with a computer!

Cutting Cards by Hand

- Not many people can pick up exactly 26 cards from a deck all the time!
  - Although it can be done, with practice!
- After a standard cut, the previous top and bottom cards are now together in the middle of the pack
  - Cheaters & magicians can exploit this fact!
- There are fancier cuts, e.g. divide the deck into 3 parts, and rearrange them
What Does Cutting Do?

• Think for a moment how cards are rearranged in a standard cut
• How does the deck change?
• Let’s look at a simplified example using only the hearts

A Neat Connection

• Cutting a deck of cards is the same as doing cut and paste in a word processor!
  – Replace cards with letters
  – Imagine highlighting the left part
  – Then “drag” it to the end of the right part

Weird Cuts: A Slow Cut

Put the bottom card of the deck on the top
Repeat the previous action 26 or so times
• People can’t do this very quickly, but computers can
  – Computers don’t move physical cards, only representations of them

Weird Cuts: Another Slow Cut

From the top of the deck, deal 26 or so cards face-up onto the table
Turn over the pile on the table
Place the pile on the table on top of the pile in your hand
• Admittedly, these two cuts won’t impress your poker buddies (unless they are CS students!)

Weird Cuts: A Flip Cut

– Hold the deck in your right hand
– Pick up the top half in your left hand
– Turn the pile in your left hand upside down
– Turn the pile in your right hand upside down
– Put the pile in your left on the pile in your right
– Turn the deck upside down (so it’s now right-side up)
Different Views of the Same Thing

- Cutting a deck of cards is the same as
  - Doing cut and paste in a word processor
  - Rotating a sequence in a circular fashion
    - This is what the slow cuts did directly
  - Swapping unequal segments of a sequence
    - This is what the flip cut did directly

Shuffling Cards

- There are many ways to shuffle cards
  - Milk shuffle: repeatedly remove the top and bottom cards
  - Riffle shuffle: cut the deck into two piles, and interleave the cards
    - A perfect riffle shuffle is when you cut the deck into two piles of 26, and perfectly alternate cards from each pile
    - Also known as a faro shuffle, or a weave shuffle

Shuffling Cards

- We will consider two sorts of shuffles:
  - A random shuffle, where the deck is randomly permuted
  - The standard riffle shuffle, where the deck is cut into two approximately equal halves, and the deck is re-formed by alternately adding cards from each half

Random Shuffle

- We want to put the deck into a completely random order
  - We want to randomly permute the deck
  - This means that each card has the same chance of being in any position
  - Ideally, we want to do this quickly, and without using extra memory

52 Pick Up

- Here’s one way to generate a random permutation by hand:
  - Throw the deck in the air so the cards fly everywhere
  - Then pick them up, in no particular order
  - Card players don’t usually shuffle cards this way, but computers often do

Faro Shuffles

- To do a faro shuffle
  - Make two piles of exactly 26 cards each
  - Alternate choosing a card from each pile, placing the selected card on a new pile
  - If the top card remains the top card, it’s called an out shuffle; otherwise it’s an in shuffle
  - 8 out shuffles in a row returns a deck to it’s original order!
Imperfect Riffle Shuffles

– Make two piles of roughly 26 cards each
– Alternate choosing clumps of, say, 0 to 3 cards from each pile, placing the clump on the new pile

• We can program a computer to do either perfect or imperfect riffle shuffles

Which Shuffle to Choose?

• If you are writing a card game, you probably want randomness
  – Perfect riffle shuffles don't mix up the cards very well
• Thus, either a random permutation or an imperfect riffle shuffle will do
• How can you create a random permutation of cards?

Random Permutations:

Method 1

• This simple method works very well:
  For i = 1 to 52 do
    Choose R to be a random number from 1 to 52
    Swap card i with card R
  End for

• It's fast, doesn't use extra memory, and is easy to understand and program
  – Hard to do by hand, though!

Random Permutations:

Method 2

– Hold the deck in your hand
– Select a random card and place it face down on a pile on the table
– Repeat the previous step until you no longer hold any cards in your hand

• This is simple, but it requires twice as much memory as the previous method
  – That's okay for only 52 cards

Pseudocode for Method 2

unprocessed contains all the cards
processed is empty
While not unprocessed.empty() do
  Remove a randomly chosen card c from unprocessed
  Put c on top of processed
End while
Representing a Permutation

- Our goal: generate random permutations of cards
- But let's look at the bare essentials first
- Let's work with a vector<int> instead of cards
  - Imagine writing the numbers 1 to 52 on your cards
- Later, you can use a Card class

Permutations of ints

- This puts 1 to 52 into a vector:
  ```cpp
  vector<int> p; // processed
  Vector<int> u; // unprocessed
  for(int i=1; i<=52; ++i)
    u.push_back(i);
  ```
- We can swap u[i] with u[j] using
  ```cpp
  swap(u[i], u[j])
  ```

Method 2 in C++

- The vectors u and p have just been correctly initialized, so here’s the loop:
  ```cpp
  while (u.size() != 0) {
    int r = 1 + (rand() % 52);
    p.push_back(u[r]);
    swap(u[r],u[u.size() - 1]);
    u.pop_back(); // remove u’s last element
  } // while
  ```

Method 2 Modified

- We can rewrite this method to so it only uses one vector
- The trick is to notice that the number of cards in processed and unprocessed is always 52
- We can partition one 52-element vector into two parts, one for processed, the other for unprocessed

The following slides show how to randomly permute a vector ...

1 2 3 4 5 6 7 8 9

Algorithm: move randomly selected numbers from the unprocessed part into the processed part, stopping when the unprocessed part is empty ...

1 2 3 4 5 | 7 8 9

Red numbers are selected randomly
swapped

unprocessed

processed

swap

unprocessed

processed

swap with itself

unprocessed

processed

swapped with itself

unprocessed

processed