Selection Sort

- Sorts the elements of an array.
- Very easy to code.
- Has TERRIBLE performance
  - “O(N^2)” “N squared”
  - It's really good when N is small!!
Selection Sort Strategy

• Select the smallest element into A[0].
  – NOW:
    • A[0] contains its final, good answer value.
    • You just have to worry about A[1], A[2], to the end.

• Ignoring A[0], apply the same strategy to the rest of the array.
Strategy articulated into loops

- Outer loop: for $X = 0, 1, 2, \text{ who knows???}$
  - select the right answer into $A[X]$
  - Inner loop does the selection job:
    - For $Y = X+1, X+2, \text{ etc to who knows ???}$
    - THINK: Let's maintain the answer so far in $A[X]$
      - //HA! We need to move value in $A[Y]$ into $A[X]$
        - //BUT we better NOT LOSE the old value in $A[X]$
      - 

Strategy articulated into loops

- Outer loop: for $X = 0, 1, 2, \ldots$ who knows???
  - select the right answer into $A[X]$
  - Inner loop does the selection job:
    - For $Y = X+1, X+2, \ldots$ to who knows ???
      - //Brilliant IDEA: **SWAP** them! Use famous swap pattern.
        - String $temp = A[X]$;
        - $A[Y] = temp$;
The outer limit

- A is A[0], A[1], ... A[length−2], A[length−1]
  
  Once A[length−2] has the right answer,
  
  (A) You STILL have to find the largest string and put it it A[length−1], otherwise there's a bug
  
  (B) You're done!

- The best choice for the for loop continuation condition: [ for (X=0; ??? ; X++) { /*inner*/ } ]
  
  (A) X < length
  
  (B) X < length − 1
The inner limit

- A is $A[0], A[1], \ldots A[length-2], A[length-1]$
  - Once $A[length-2]$ has the right answer,
    - (B) You're done, since the last $A[]$ has the biggest..
- The best choice for the inner loop continuation condition:
  
  ```java
  for (X=0; X<length-1 ; X++) {
      for(Y = X+1; Y<??; Y++) if(...) {swap}  }
  ```
  - (A) $Y < length$
  - (B) $Y < length - 1$