Aula 22 (10/Jan) - Dados mutáveis. Pesquisa e ordenação.

Módulos: parte 5 (5.4)

Sumário
Dados mutáveis.
Pesquisa sequencial e pesquisa binária.
Algoritmos de ordenação: selection sort, bubble sort e quick sort.

- Pesquisa sequencial

  pesquisa_seq.scm

```
(define teste (vector 9 7 2 8 5 3 1 6))

(define pesquisa-seq
  (lambda (vec elem)
    (letrec ((aux
              (lambda (i)
                (if (< i (vector-length vec))
                    (if (equal? elem (vector-ref vec i))
                        i
                        (aux (add1 i)))
                    #f))))
      (aux 0))))
```

- Pesquisa binária

  pesquisa_bin.scm

```
(define teste (vector 1 3 4 5 7 9))

(define pesquisa-bin
  (lambda (vec elem)
    (letrec ((aux
              (lambda (ini fim)
                (if (> ini fim)
                    #f
                    (let ((meio (quotient (+ ini fim) 2)))
                      (cond
                        ((equal? elem (vector-ref vec meio)) meio)
                        ((< elem (vector-ref vec meio)) (aux ini (sub1 meio)))
                        (else (aux (add1 meio) fim))))))))
      (aux 0 (sub1 (vector-length vec))))))
```

- Ordenação por selecção (selection sort)

  selectionsort.scm

```
(define teste (vector 8 6 3 6 8 2 8 9 3 5 7 3 5 4 3 1))

(define selection-sort!
  (lambda (vec)
    (letrec ((aux-min
              (lambda (i minimo pos-min)
                (if (< i (vector-length vec))
                    (if (< (vector-ref vec i) minimo)
                        (aux-min (add1 i) (vector-ref vec i) i)
                        (aux-min (add1 i) minimo pos-min))
                    pos-min)))
             (aux-linha
              (lambda (i)
                (if (< i (sub1 (vector-length vec)))
                    ; percorre para cada valor
                    (begin
                      (let* ((base (vector-ref vec i))
                             (pos-min (aux-min (add1 i) base i)))
                        (if (not (= i pos-min))
                            ; troca valores
                            (begin
                              (vector-set! vec i (vector-ref vec pos-min))
                              (vector-set! vec pos-min base))))
                      (aux-linha (add1 i)))))))
      (aux-linha 0))))

teste
(selection-sort! teste)
```

```
teste
```

- Ordenação por bolha (bubble sort) - melhor para vectores “quase ordenados”

  bubblesort.scm

```
(define teste (vector 8 6 3 6 8 2 8 9 3 5 7 3 5 4 3 1))
;(define teste (vector 9 2 3 3 3 3 4 5 5 6 7 7 7 8))

(define bubble-sort!
  (lambda (vec)
    (letrec ((aux-bolha
              (lambda (i ordenado)
                (if (>= i (vector-length vec))
                    ordenado
                    (if (< (vector-ref vec i)
                           (vector-ref vec (sub1 i)))
                        (begin  ; troca
                          (let ((temp (vector-ref vec i)))
                            (vector-set! vec i (vector-ref vec (sub1 i)))
                            (vector-set! vec (sub1 i) temp))
                          (aux-bolha (add1 i) #f))
                        (aux-bolha (add1 i) ordenado))))))
      (if (not (aux-bolha 1 #t))
          (begin (display ".")
                 (bubble-sort! vec))
          (newline)))))

teste
(bubble-sort! teste)
teste
```

- Ordenação rápida (quick sort)

  quicksort.scm

```
(define teste (vector 8 6 3 6 8 2 8 9 3 5 7 3 5 4 3 1))

(define quicksort!
  (lambda (vec)
    (letrec ((aux-particao
              (lambda (ini fim pivot)
                (if (> ini fim)
                    ini
                    (cond
                      ((< (vector-ref vec ini) pivot) (aux-particao (add1 ini) fim pivot))
                      ((> (vector-ref vec fim) pivot) (aux-particao ini (sub1 fim) pivot))
                      (else ;troca
                       (let ((temp (vector-ref vec ini)))
                         (vector-set! vec ini (vector-ref vec fim))
                         (vector-set! vec fim temp))
                       (aux-particao (add1 ini) (sub1 fim) pivot))))))
             (aux
              (lambda (ini fim)
                (if (< ini fim)
                    (let ((pivot-pos (aux-particao ini fim (vector-ref vec (quotient (+ ini fim) 2)))))
                      (aux ini (sub1 pivot-pos))
                      (aux pivot-pos fim))))))
      (aux 0 (sub1 (vector-length vec))))))

teste
(quicksort! teste)
teste
```